University of Alberta

Reduced Models of Software Development Effort Estimation

by

Zhenyou Li

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of
Master of Science

Department of *Electrical and Computer Engineering*

Edmonton, Alberta

Spring 2004

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-612-96513-9
Our file Notre référence
ISBN: 0-612-96513-9

The author has granted a non-exclusive license allowing the Library and Archives Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou aturement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

Estimating the amount of effort required for developing a software system is one of the most important project management concerns. This thesis has thoroughly investigated the multiple regression and neural networks in effort estimation, built efficient reduced models while the predicting accuracy is guaranteed, and identified the most significant factors among independent variables.

It has been found that neural networks used as effort estimators are more accurate but less repeatable, while multiple regression is less accurate but repeatable.

When building more efficient reduced models, linear techniques and genetic algorithms are employed for feature set selection; linear techniques are straightforward but less efficient especially when the number of dataset attributes is large. Genetic algorithms are more efficient for feature set selection. The most significant factors identified in the optimal reduced model are especially important for managers in making decisions in the early stage of software development.

Dedication

To my son Yuanshan "Sam", whom was born in Edmonton, Alberta, Canada In October, 2003.

To my wife Hua Cai.

Chapter 1 Introduction

1.1 Literature Research

1.2 Proposed Research Work

1.3 Methods of the Proposed Research

1.4 Summary

1.5 Bibliography

Chapter 2 Multiple Regression Models and Reduced Models

2.1 Introduction

2.1 Methodology

2.3 Results and Analysis

2.3.1 Applying Multiple Regression Directly on the COCOMO Dataset

2.3.2 Applying Multiple Regression Indirectly on the COCOMO Dataset

2.3.3 Detection of Potential Outliers and the Influential Effects on the Reduced Regression Model

2.3.4 Detection of Potential Outliers and the Influential Effects on the Full Regression Model

2.3.5 Variable Selection Results Using Forward Selection

2.3.6 Variable Selection Results Using Backward Elimination

2.3.7 Variable Selection Results Using Stepwise Method

2.3.8 Variable Selection Results Using Maximum R^2 Improvement

2.3.9 Variable Selection Results Using Minimum R^2 Improvement

2.3.10. Final Optimal Reduced Regression Model

2.4 Summary

2.5 Bibliography

Chapter 3 Using Neural Networks for Effort Estimation Modeling

3.1 Introduction

3.2 Methodology

3.3 Experimental Results and Analysis

3.3.1 Using Linear Transfer Function in Output Layer and Unipolar Sigmoid Transfer Function in Hidden Layer

3.3.2 Simulation Using Categorical Data instead of Numerical Data

3.3.3 Comparisons of Various Simulations with Optimal Parameters

3.4.Summary

3.5 Bibliography

Chapter 4 Feature Optimization Guided by Genetic Algorithms

4.1 Introduction

4.2 Methodology for Real-valued Coded Genetic Algorithms

4.3 Experimental Results and Analysis

4.3.1 Using Multiple Linear Regression to Evaluate Fitness

4.3.2 Using Neural Networks to Evaluate Fitness

4.3.3 Comparisons of Linear Methods and Genetic Algorithms on Feature Optimization

4.4 Summary

4.5 Bibliography

Chapter 5 Conclusions and Future Work

5.1 Conclusions

5.1.1 Multiple Regression Models

5.1.2 Neural Networks Simulations

5.1.3 Feature Selection and Optimization Using Linear and Non-linear Search Methods

5.2 Future Work

5.3 Bibliography

Appendices

List of Tables

Table 1-1 Coefficients determined by development modes

Table 2-1. Pearson's Correlation Matrix between the 16 Explanatory Variables and the Dependent Variable Effort

Table 2-2. Full Multiple Regression Model of the Dependent Variable Effort and the 16 Explanatory Variables

Table 2-3. Pearson's Correlation Matrix between the 15 Explanatory Variables plus ln(size) and the Dependent Variable ln(Effort)

Table 2-4. Full Multiple Regression Model of the Dependent Variable ln(Effort) and the 15 Explanatory Variables plus ln(size)

Table 2-5. Reduced Multiple Regression Model of the Dependent Variable ln(Effort) and the 6 Explanatory Variables plus ln(size)

Table 2-6. Influence Diagnostics for Reduced Model on the COCOMO Dataset

Table 2-7. Results of Forward Selection Procedure for Dependent Variable ln(Effort)

Table 2-8. Summary of Forward Selection

Table 2-9. The Optimal Reduced Regression Model for Forward Selection Procedure

Table 2-10. Results of Backward Elimination Procedure for Dependent Variable ln(Effort)

Table 2-11. Summary of Backward Elimination Procedure

Table 2-12. Optimal Reduced Regression Model for Backward Elimination Procedure

Table 2-13. Results of Stepwise Selection Procedure for Dependent Variable ln(Effort)

Table 2-14. Summary of Stepwise Selection

Table 2-15. Results of Maximum R^2 Improvement Procedure for Dependent Variable ln(Effort)

Table 2-16. Final Reduced Regression Model of Maximum R^2 Improvement Procedure for Dependent Variable ln(Effort)

Table 2-17. Summary of the Minimum R^2 Improvement Method

Table 2-18. Optimal Reduced Regression Model

Table 3-1. Comparison Performance of Networks with Bipolar and Unipolar Sigmoid as Transfer Functions

Table 3-2. Effects of Learning Rates and Hidden Nodes on RMSE Using Linear Transfer Function in Output Layer

Table 3-3. The Effects of Number of Hidden Node on RMS at Constant LR

Table 3-4. Effects of Momentum on the Networks Performance Using Linear Transfer Function in Output Layer at LR=0.02

Table 3-5 The Effects of λ on RMSE at Constant LR

Table 3-6 Relationships between Numerical and Categorical Input Variables for Appendix 3

Table 3-7 Comparison of Performance of Networks with Various Transfer Functions

Table 4-1. Optimal Subset of Variables by GA Using Multiple Regression to Evaluate Fitness

Table 4-2. Optimal Subset of Variables by GA Using NN to Evaluate the Fitness

Table 4-3. Typical Effort Estimation Using Different Models

Table 4-4. Comparison of Performance of Different Models

List of Figures

Figure 2-1. ln_Effort vs. ln_size Plot of the Reduced Multiple Regression Model with 9 Explanatory Variables

Figure 2-2. Regression Estimated Effort vs. Actual Effort by the Reduced Multiple Regression Model with 9 Explanatory Variables

Figure 2-3. ln_Effort vs. ln_size Plot of theFull Multiple Regression Model with All 16 Explanatory Variables

Figure 2-4. Regression Estimated Effort vs. Actual Effort by the Full Multiple Regression Model with All 16 Explanatory Variables

Figure 3-1. Learning Process with Bipolar Sigmoid Functions as Transfer Functions at LR=0.015

Figure 3-2. NN-Predicted EAF VS Actual EAF with Bipolar Sigmoid Functions as Transfer Functions after Epoch = 10000

Figure 3-3. Learning Process with Bipolar Sigmoid Functions as Transfer Functions (Number of Hidden Node = 5)

Figure 3-4. NN-Predicted Efforts VS Actual Efforts with Bipolar Sigmoid Functions as Transfer Functions (Number of Hidden Node =5)

Figure 3-5. Learning Process with Unipolar Sigmoid Functions as Transfer Functions (Number of Hidden Node =5)

Figure 3-6. NN-Predicted Efforts VS Actual Efforts with Unipolar Sigmoid Functions as Transfer Functions (Number of Hidden Node = 5)

Figure 3-7. RMSE VS Number of Hidden Nodes at LR=0.02

Figure 3-8. RMSE VS Learning Rate Constants at Number of Hidden Nodes =5

Figure 3-9 Learning Process of Categorical Data Processing Networks with Linear Function as Output Layer Transfer Function at LR=0.02

Figure 3-10 NN-Predicted Efforts VS Actual Efforts of Categorical Data Processing Networks with Linear Function as Output Layer Transfer Function at LR=0.02

Figure 4-1 Flowchart of basic genetic algorithm

Figure 4-2 Sample of chromosome represented in real valued coded in [0,1]

Figure 4-3 Sample of Chromosome with Optimal Subset of 8 Variables (Gene's Indexes) from Figure 4-2

Figure 4-4 Samples of Chromosomes with Corresponding Fitness Values in a Pool

Figure 4-5 Simple Crossover Operation

Figure 4-6 Mutation Operation

Figure 4-7 Typical Relations of Best Fitness VS Generations of GA Using Multiple Regression as Fitness Function

Figure 4-8 Typical Relations of Average Fitness VS Generations of GA Using Multiple Regression as Fitness Function

Figure 4-9 Typical Relations of Estimated Efforts VS Actual Efforts Using Reduced Multiple Regression With Optimal Subset of 9 Inputs Guided by GA

Figure 4-10 Typical Relations of Estimated Efforts VS Actual Efforts Using Full Multiple Regression With 16 Explanatory Variables

Figure 4-11 Typical Relations of Best Fitness VS Generations by GA Using Neural Networks as Fitness Function

Figure 4-12 Typical Relations of Average Fitness VS Generations by GA Using Neural Networks as Fitness Function

Figure 4-13 Typical Relations of Estimated Efforts VS Actual Efforts Using Reduced NN Simulation With Optimal Subset of 9 Inputs Found by GA-NN

Figure 4-14 Typical Relations of Estimated Efforts VS Actual Efforts Using Reduced NN Simulation With Optimal Subset of 9 Inputs Obtained by GA-Multiple Linear Regression

Acknowledgement

I am very grateful to NSERC and ASERC for their financial support to our research. I would like to express many thanks to my supervisors, Dr. Petr Musilek and Dr. Witold Pedrycz, for their patient supervisions and valued suggestions; thanks a lot to my committee members Dr. Jozef Szymanski and Dr. Maret Reformat, for their precious time and efforts in reviewing my thesis.

Thank you very much to Mr. Yeon, Hean Chin and Mr. Lee, Beng Ee for proof reading; I would also like to appreciate the sincere encouragement of Mr. Li, Yifan and Mr. Su, Xiaoyuan.

My special thanks goes to my wife, Hua Cai, without her understanding, tolerance and support, I cannot finish this thesis.

Chapter 1 Introduction

1.1. Literature Research

The principal components of project costs consist of hardware costs, travel and training costs, and effort costs (the costs of paying software engineers, usually measured in man-months or man-hours). This thesis focuses on the effort costs estimation, especially in the early development stages. In the beginning of a new software project, the project manager needs to be able to estimate the effort cost and evaluate how accurate the estimation is. This is a crucial but difficult task. Therefore, it requires significant effort to correctly estimate software project development effort. However, in real world, this process is often neglected. Until now there is still no simple way to accurately estimate software development effort.

Most people working in the software industry recognize that developing software to predictable schedules is a risky business. Some data in a 1995 study by the Standish Group showed that on average only 16% of software projects are completed successfully (all specified features delivered on-time and budget). 31% were canceled, and the remaining 53% projects were dramatically over budget and schedule, and delivered less functionality than originally specified. While the reasons for these failures and near failures are diverse, the most common reasons are:

- Specifying incomplete or unclear requirements
- Failing to adjust schedules when scope changes
- Setting development schedules that are too aggressive
- Insufficient resources.

These failures are due to lack of data and poor effort estimation. Effective software estimation combined with sound project management discipline can solve many of these problems.

The size of a software project in terms of the development effort and schedule can be estimated using mathematical models based on data from software projects that have already been completed. The results from these models can be refined using various tuning parameters such as technical complexity of the product, experience of the development team, and maturity of the development process. The results of the estimation process will provide project managers with a range of options and probabilities for the likely project timelines and levels of effort required to achieve them. Armed with this piece of information, project managers are able to make recommendations to senior management and help minimize the project risk by setting realistic development schedules, and making appropriate feature tradeoff. However, software estimation is not an exact science, therefore, it is important to refine the estimate and the project schedule as the product becomes more explicitly defined during the requirements and design stages.

Effective effort estimation is one of the most important software development activities, however, it is also one of the most difficult. Underestimating a project will lead to understaffing, insufficient quality assurance, and too short a schedule. That in turn can lead to staff burnout, low quality, loss of credibility as deadlines are missed, and inefficient development effort. Overestimating a project is equally bad. According to Parkinson's Law, work expands to fill available time. As a result, projects that can be

completed in a much shorter time frame will take longer to complete if more time is given. An accurate and robust estimate is a critical part of an efficient software project.

Software engineering studies found that software projects effort estimated using automated estimation software are more accurate than that estimated using manual methods. Estimates based on historical data from an organization are more accurate than estimates based on rules of thumb or educated guesswork. Automated estimation tools often allow software projects to be delivered at lower cost than manual methods do.

Despite extensive research efforts to understand the factors contributing to the success of software projects [3,4,5], many software development projects are still failing and such research is likely to continue until the development process is under better control [4]. For example, in 1994, 31% of all corporate software development projects resulted in cancellation [14]. A more recent study found that 20% of software projects failed, and 46% experienced cost and schedule overruns or products with significantly reduced functionality [15]. Troubled projects can cause developers to suffer long hours of unpaid overtime, lack of motivation, and burnout, leading to excessive staff turnover and its associated costs.

More comprehensive review of different techniques for effort estimation has also been discussed in the literatures [16,17,18]. The popular approaches include: algorithmic and parametric models, expert judgment, formal and informal reasoning by analogy, price-to-win, top-down, bottom-up, rules of thumb, and available capability. According to Boehm [1], three acceptable methods can be distinguished: the expert method, the analogy method, and the use of software cost estimation models.

1.1.1 Algorithmic and Parametric Models

There have been a number of estimation models produced over the past three decades. These efforts can be classified into (1) model-based techniques, (2) expertise-based techniques, (3) learning-oriented techniques, and (4) dynamic techniques [10]. Most software development effort estimators are model-based with well-known examples including COCOMO [1], function points analysis (FPA) [11] and SLIM [12]. SLIM uses linear programming to consider the constraints of development effort, but its estimates are very sensitive to technical factors, and it is not suitable for small projects. FPA estimates need only the detailed specification of the early stage, it is more accurate than the estimates of lines of source code since it is independent of language and layout, but FPA counting is quite subjective and it is hard to automate. In model-based techniques, the development effort is defined as a function of variables representing the most important cost drivers of the project.

In the software engineering industry, the Constructive Cost Model (COCOMO) introduced by Barry Boehm [1] for software effort estimation is one of the most popular project management tools. The main focus in COCOMO is on estimating the effects of the 15 cost drivers, together with software size, on development effort. COCOMO is a collection of three sub models: basic, intermediate and detailed. The intermediate model is of interest in this thesis. Its predictions of required effort (measured in Person-Months – PM) are based primarily on the estimate of the software project's size (measured in Thousands of Source Lines of Code – KSLOC):

$$Effort = A * EAF * (Size)^{B} \qquad (1\text{-}1)$$

Where *EAF* is the Effort Adjustment Factor derived from the 15 cost drivers, and *A* and *B* are constants derived from the development mode and application type.

There are three development modes: organic, semi-detached, and embedded. *Organic mode* is used when the constraints upon project development are mild and the given project has been preceded by a number of similar projects that could assist in defining the agenda of development. *Embedded mode* is used for projects that have very tightly defined constraints, These projects cannot rely on previously completed projects. *Semi-detached mode* is used for projects where constraints are greater than the organic mode, but there still remains some flexibility. These projects may only be preceded by a few similar projects. The relationship between development modes and the coefficients is illustrated below.

Table 1-1 Coefficients determined by development modes

Development Mode	A	B
Organic	3.2	1.05
Embedded	2.8	1.20
Semi-detached	3.0	1.12

In COCOMO model, there are 15 cost drivers under the four main categories [1]. Taking software size into consideration, there are altogether 16 attributes needed for effort estimation. COCOMO is transparent, and the estimator can see how it works, the drivers are particularly helpful for the estimator to understand the impacts of the drivers on the effort. However, the estimate models need to be calibrated to each individual measurement environment, and the estimate accuracy is still variable even after

calibration, it is very hard to estimate the software size in the early stage of development, it is also extremely vulnerable for managers to misclassify the development mode of the new project, the success of estimation heavily depends on the historical data, which is not available in the early stage of project development. Moreover, it is quite possible that some of these cost driver factors may be correlated when contributing to the effort estimation. This correlation may even affect the accuracy of the model.

1.1.2 Learning-Oriented Estimation

More recently, machine learning models have been developed in the area of cost estimation. In [19,20,21,22,23], a Case-Based Reasoning (CBR) approach is adopted in constructing a cost model for the latter stages of the development life cycle. Delany [24] also uses a CBR approach applied early in development life cycle.

Artificial neural network (ANN) is the most commonly used learning-oriented estimator for estimating software development effort [13], and back-propagation is the most commonly used learning algorithm in ANN. Flexibility, objectivity, correctness and computational economy are the desirable features that make this combination attractive for data modeling application. An overview of the ANN with back-propagation algorithm is described in [5].

Srinivasan [25] builds a variety of models including neural networks, regression trees, COCOMO, and SLIM. The training set consists of COCOMO data (63 projects from different applications). The training models are tested against the Kemerer's COMOMO data (15 projects, mainly business applications). The regression trees outperformed the

COCOMO and the SLIM model; the neural networks and function point-based prediction models outperformed regression trees.

Samson [26] applies neural network models to predict effort from software sizing using COCOMO-81 data, the neural network models produced better results than COCOMO-81. Wittig et al. [27] estimated development effort using a neural network model. They achieved impressive results of 75 percent accuracy pred(25). Boetticher [28] conducted more than 33,000 different neural network experiments on empirical data collected from separate corporate domains. The experiments assessed the contribution of different metrics to programming effort. The research produced a cross-validation rate of 73.26% in terms of pred (30). Hodgkinson [29] adopted a top-town approach using neurofuzzy cost estimator in predicting project effort, the results were comparable to other techniques including least-square multiple linear regressions, estimation via analogy, and neural networks.

1.1.3 Expert Judgment

Expert judgment is widely used as an estimation method. It involves consulting with one or more experts. One or more experts in both software development and the application domain use their experience to predict software costs. Process iterates until some consensus is reached. The advantages are that it is relatively inexpensive and may be accurate if the experts have direct experience in similar systems. The disadvantage is that it can be very inaccurate if there is no or few real expert.

1.1.4 Estimation by Analogy

Estimation by analogy involves reasoning by analogy, using experience with one or more completed projects to relate actual cost and development time to the cost and development time of the new project. The effort of a project is computed by comparing the project to a similar project in the same application domain. The results can be accurate if sufficient project data is available; however, it is impossible to employ this method if no comparable project has been tackled before, and it also needs a systematically maintained cost database.

1.2. Proposed Research Work

From the literatures above, each method has its strengths and weaknesses; estimation should be based on several methods. Pricing to win is sometimes the only applicable method. If these do not return approximately the same result, and there is insufficient information available, some action should be taken to find out more in order to make more accurate estimates. Poor effort estimation is the most significant factor to cause the failure, and data scarcity in the early stage of software development is the main cause of poor effort estimation. The recent research is focused on machine learning methods, especially neural networks, which have the abilities to deal with domain complexity and to generalize, along with adaptability, flexibility and parallelism. The goals of this thesis is:

Accurately predict the efforts required for a new software development process.

This is the first step; accuracy should be the fundamental requirements for all the estimation models, otherwise, estimated results are not reliable.

Building reduced efficient models and identifying the major factors contributing to the software development estimation.

Identifying the most significant factors will be helpful for managers to make decisions in the early stages of software development, as proposed in the following sections, there are many successful methods to build such reduced models. Software managers would benefit from a reduced model considering only the most significant factors contributing to the success of the software systems, especially in the early stage of development. At the same time, such reduced model must remain accurate, since its major function is to predict the effort needed for new software projects. The goal of this thesis is to provide a reduced yet accurate model for effort estimation in the second step. To achieve this goal, the most significant attributes must be identified among the attributes of the original COCOMO model using data mining methods [6,7,8,9].

1.3. Methods of the Proposed Research

1.3.1 Model Reduction

The major factors contributing to the success of software systems fall into seven categories [3,4]: (1) management, (2) customers and users, (3) requirements, (4) estimation, scheduling, (5) project manager, (6) software development process, and (7) development personnel. The approach described in this thesis is based on four categories adopted from [1], which is summarized in following: (1) project attributes, (2) personnel attributes, (3) computer attributes, and (4) product attributes. These two classification systems are actually overlapping to some extent. For example, the project manager and development personnel in the first system can be classified into personnel attributes in the latter system, which also includes the analysts and programmers aside from project managers and development personnel.

There are several multiple regression methods for building efficient reduced linear models that can be used for effort prediction. Based on the full regression model that associates the dependent variable effort to all 16 explanatory variables in the COCOMO data, efficient reduced models can be devised using methods such as forward selection, backward elimination, stepwise selection, stepwise selection with maximum R^2 improvement, and minimum R^2 improvement. All these variable selection methods are only suitable for building linear models.

1.3.2 Evaluating Performance of Models

A key factor in selecting a software estimation model is the accuracy of its predictions. In order to compare the performance of multiple regression models with that of the NN simulation, the mean magnitude of relative error (*MMRE*) and *Pred(25)* are employed to determine the effort prediction accuracy. The *MMRE* is determined as:

$$MMRE = \frac{100}{N}\sum_{i=1}^{N}\left(\frac{\left|Effort_a^i - Effort_e^i\right|}{Effort_a^i}\right) \quad (1\text{-}2)$$

where $Effort_a$ is the actual effort, and $Effort_e$ is the estimated effort. The lower the *MMRE*, the more accurate the estimation. *Pred(25)* indicates the percentage of predictions with error less than 25 percent of the actual value. The higher the *Pred(25)*, the more accurate the estimation is.

1.3.3 Real Value Coding Genetic Algorithms

Genetic Algorithms (GAs) may deal successfully with a wide range of domains. The reasons for this success are: 1). GAs can solve hard problems quickly and reliably, 2) GAs are easy to interface with existing simulations and models, 3) GAs are extensible and 4) GAs are easy to hybridize.In other words, GAs are robust.

The basic genetic algorithm is as follows:

1. **[Start]** Generate random population of *n* chromosomes (suitable solutions for the problem)
2. **[Fitness]** Evaluate the fitness *f(x)* of each chromosome *x* in the population
3. **[New population]** Create a new population by repeating following steps until the new population is complete

3a. [Selection] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)

3b. [Crossover] With a crossover probability cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.

3c. [Mutation] With a mutation probability mutate new offspring at each locus (position in chromosome).

3d. [Accepting] Place new offspring in the new population

4. **[Replace]** Use new generated population for a further run of the algorithm
5. **[Test]** If the end condition is satisfied, **stop**, and return the best solution in current population
6. **[Loop]** Go to step **2**

1.4. Summary

To accurately predict the software development effort is still the most difficult and important task for project managers. This thesis has reported the current situations, the solutions/suggestions, their advantages and disadvantages in software effort estimation, and tries to identify the most significant factors contributing to the success of software development effort estimation in the early life cycle. According to previous research, machine learning methods especially neural networks and genetic algorithms are the promising methods to solve the difficulties in data scarcity and thus improve the effort

estimation. The proposed methods should be accurate, simple to use, more efficient in software development effort estimation.

1.5 Bibliography

[1] B. W. Boehm, Software Engineering Economics, Prentice Hall, 1981.

[2] J. D. Jobson, Applied Multivariate Data Analysis, Volume 1: Regression and Experimental Design, Springer, New York, 1991.

[3] J. Drew et al. *Case Study: factors for early prediction of software development success,* Information and Software Technology, 44 (2002) 53-62.

[4] B. A. Kitchenham, *The question odd scale economies in software—why cannot researchers agree?* Information and Software Technology, 44 (2002) 13-24.

[5] A. Heiat, *Comparison of artificial neural network and regression models for estimating software development effort.* Information and Software Technology, 44 (2002) 911-922.

[6] T.P. Ryan, Modern Regression Methods, Join Wiley & Sons, Inc. New York, 1997.

[7] H. Liu and H. Motoda, Feature Selection for Knowledge and Data Mining, Kluwer Academic Publishers, 1998.

[8] K. Cios, W. Pedrycz and R. Swinarski, Data Mining Methods for Knowledge Discovery, Kluwer Academic Publishers, 1998.

[9] H. Liu and H. Motoda, Instance Selection and Construction for Data Mining, Kluwer Academic Publishers, 2001.

[10] B. Barry, C. Abts, S. Chulani, Software development cost estimation approaches—a survey, Annals of Software Engineering February (2000).

[11] A. J. Albrecht, J. Gaffney, Software function, source lines of code, and development effort prediction: a software science validation, IEEE Transaction on Software Engineering 9(6) (1983) 639-648.

[12] L. Putnam, W. Myers, Measures foe Excellence, Yourdon Press Computing Series, 1992.

[13] A. Gray, S. McConnell, *A comparison of techniques for developing predictive models of software metrics*, Information and Software Technology (1997)39.

[14] S. McConnell, Rapid Development, Microsoft Press, Redmond, 1996.

[15] R. L. Glass, How not to prepare for a consulting assignment and other ugly consultancy, Communications of the ACM, 41 (12), 1998.

[16] B. Boehm et al., Cost models for future software life cycle process: COCOMO 2, Annals of software engineering, 1995.

[17] F Heemstra. Software cost estimation, Information and Software Technology, October 1992.

[18] J. Hihn, H. Habib-Agahi, Cost Estimation of Software Intensive Projects: A Survey of Current Practices, Proceedings of International Conference on Software Engineering, 1991.

[19] R. Bisio, F. Malabocchia, Cost Estimation of Software Projects Through Case-Based Reasoning. Case-Based Reasoning Research and Development. First International Conference, ICCBR-95 Proceedings, 1995.

[20] G. Finnie et al., Estimating Software Development Effort with Case-based Reasoning. Proceedings of International Conference on Case-Based Reasoning, D. Leake, E Plaza, (Eds), 1997.

[21] G. Kadoda et al., Experiences Using Case-Based Reasoning to Predict Software Project. Effort. Empirical Software Engineering Research Group Technical Report, Bournemouth University, January 27, 2000.

[22] T. Mukhopadhyay, and S. Kekre, Software Effort Models for Early Estimation of Process Control Applications, IEEE Transactions on Software Engineering, 18 (10 October), 1992.

[23] M. Prietula et al., Software Effort Estimation with a Case-Based Reasoner, Journal of Experimental and Theoretical Artificial Intelligence, 8(3-4), 1996.

[24] S. J. Delany and P. Cunningham, The Application of Case-Based Reasoning to Early Project Cost Estimation and Risk Assessment. Department of Computer Science, Trinity College Dublin, TDS-CS-2000-10, 2000.

[25] K. Srinivasan and D. Fisher, Machine Learning Approaches to Estimating Software Development Effort, IEEE Trans. Software Engineering, February 1995.

[26] B. Samson et al., Software Cost Estimation Using An Albus Perceptron, Information and Software Technology, 1997.

[27] G. Wittig and G. Finnie, Estimating Software Development Effort with Connected Models. Information and Software Techology, 1997.

[28] G. Boetticher, An Assessment of Metric Contribution in the Construction of a Neural Network-Based Effort Estimator, Second Int. Workshop on Soft Computing Applied to Soft. Engineering, 2001.

[29] A.C. Hodgkinson and P.W. Garratt, A Neurofuzzy Cost Estimator, Proceedings 3rd International Conference in Software Engineering and Applications (SAE), 1999.

Chapter 2 Multiple Regression Models and Reduced Models

2.1 Introduction

Single regression techniques use two variables to predict a best fit line for a certain set of data. This is not always reliable as there are often many conditions that should be accounted for when drawing the best fit line. Multiple regression can be used to account for these extra variables. The advantages of using multiple regression are numerous. If more relevant data is included, a more accurate and reliable prediction can be made.

Multiple regression shares all the assumptions of other types of correlation. The relationships between variables have to be linear. Other types of regression can be used to determine quadratic, exponential, cubic, and other forms, but are beyond the scope of this thesis. There must be the same level of relationship throughout the range of the independent variables. Finally, each variable should be chosen to fit the model being tested. The exclusion of important contributing variables or the inclusion of extraneous variables can change the best fit line and provide erroneous predictions.

We can expect that using multiple regression will provide an accurate equation to represent the COCOMO dataset. Using more variables decreases the chance for error, and increases the reliability of the equation itself.

Two practical considerations that should be noted are the choice of the number of variables and the importance of residual analysis. The number of variables used can significantly affect the outcome of the regression model. Each variable must be chosen carefully to make certain that the variable is not extraneous. Also, with the addition of more variables, more observations need to be obtained so that the estimates of the regression line are probably stable and likely to replicate.

2.2 Methodology

Multiple regression methods will be employed to build the regression models for predicting the projects' effort: first build the full regression model with all 16 explanatory variables relating to the dependent variable effort and then based on that, build some efficient reduced models. The techniques for building reduced models include linear procedures such as forward selection, backward elimination, stepwise selection, Maximum R^2 improvement, and Minimum R^2 improvement.

2.2.1 Overview of Multiple Linear Regression

A multiple linear regression problem can be stated as follows:

If the x_j are varied and the n values $Y_1, Y_2, ..., Y$ of Y are observed, then we write:

$$Y_i = \beta_0 + \beta_1 x_{i1} + ... + \beta_{p-1} x_{i,p-1} + \varepsilon_i (i = 1,2,...,n) \qquad (2\text{-}1)$$

where x_{ij} is the i^{th} value of x_j. Writing these n equations in matrix form we have:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \cdots \\ Y_n \end{bmatrix} = \begin{bmatrix} x_{10} & x_{11} & \cdots & x_{1,p-1} \\ x_{20} & x_{21} & \cdots & x_{2,p-1} \\ \cdots & \cdots & \cdots & \cdots \\ x_n & x_{n1} & \cdots & x_{n,p-1} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \cdots \\ \beta_{p-1} \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \cdots \\ \varepsilon_n \end{bmatrix}$$

or:

$$Y = X\beta + \varepsilon \qquad (2\text{-}2)$$

where $x_{10} = x_{20} = ... = x_{n0} = 1$

We call the n x p matrix X the *regression matrix*, each Y_i a *response variable*, Y the *response vector*, and x_j the *predictor variable*.

Parameter Calculation by Least Squares Minimization

The *method of least squares* consists of minimizing $\sum_i \varepsilon_i^2$ with respect to β. Setting $\theta = X\beta$, it is minimized:

$$\varepsilon'\varepsilon = \| Y - \theta \|^2 \qquad (2\text{-}3)$$

subject to:

$$\theta \in \Re[X] = \{y : y = X\beta\} \qquad (2\text{-}4)$$

Let $\hat{\beta}$ be the least squares estimate of β. The fitted regression $X\ \hat{\beta}$ is denoted by:

$$\hat{Y} = [(\hat{Y}_i)] \qquad (2\text{-}5)$$

The elements of $e = [Y - X\beta]$ are called the *residuals*. The value of:

$$RSS = e'e = [Y - X\beta]'(Y - X\beta) \qquad (2\text{-}6)$$

is called the *residual sum of squares*. The matrix:

$$\mathbf{R} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1,p-1} \\ x_{21} & x_{22} & \cdots & x_{2,p-1} \\ \cdots & \cdots & \cdots & \cdots \\ x_{n1} & x_{n2} & \cdots & x_{n,p-1} \end{bmatrix}$$

which is the regression matrix without the first column of 1s, is called the *predictor data matrix*.

Model Variance

The *variance* of the model is defined to be the variance of ε. The statistic:

$$S^2 = \frac{RSS}{n - p} \qquad (2\text{-}7)$$

is an unbiased estimator of this variance.

Parameter Dispersion (Variance-Covariance) Matrix

The *dispersion matrix* for the parameter estimates $\hat{\beta}$ is the matrix

$$D(\hat{\beta}) = (\text{cov}[\hat{\beta}_i, \hat{\beta}_j]) \qquad (2\text{-}8)$$

where $\text{cov}[\hat{\beta}_i, \hat{\beta}_j]$ is the covariance of $\hat{\beta}_i$ and $\hat{\beta}_j$. The dispersion matrix is calculated according to the formula:

$$D(\hat{\beta}) = S^2(X'X)^{-1} \qquad (2\text{-}9)$$

where S^2 is the estimated variance, as defined above, and X and X' are the regression matrix and its transpose, respectively.

Significance of the Model (Overall F Statistic)

The *overall F statistic* is a statistic for testing the null hypothesis $\beta_1 = ... = \beta_{p-1} = 0$. It is defined by the equation:

$$F = \frac{\sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})^2 /(p-1)}{\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2 /(n-p)} \qquad (2\text{-}10)$$

where $\bar{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$

This statistic follows an F distribution with *(p-1)* and *(n-p)* degrees of freedom.

p-Value of the Model

The *p-value* is the probability of seeing the value of the F statistic for a given linear regression if the *null* hypothesis:

$$\beta_0 = \beta_1 = ... = \beta_{p-1} = 0 \qquad (2\text{-}11)$$

is true.

Critical Value of the Model

The critical value of the F statistic for a specified significance level, α, is the value, v, of the F statistic such that if the F statistic calculated for the multiple linear regression is greater than v, we reject the hypothesis

$\beta_0 = \beta_1 = ... = \beta_{p-1} = 0$ at the significance level α.

Significance of Predictor Variables

Let $\hat{\beta}_j$ be the estimate for element j of the parameter vector β. The T statistic for the parameter estimate $\hat{\beta}_j$ is a statistic for testing the hypothesis that $\hat{\beta}_j$=0. It is calculated according to the formula:

$$T = \frac{\hat{\beta}_j}{\sqrt{D(\hat{\beta})_{jj}}} \qquad (2\text{-}12)$$

where $D(\hat{\beta})_{jj}$ is the j^{th} diagonal element of the dispersion matrix. This statistic is assumed to follow a T distribution with n-p degrees of freedom.

p-Values of Predictor Variables

The p-value for each parameter estimate $\hat{\beta}_j$ is the probability of seeing the value of the calculated parameter using the formula $T = \frac{\hat{\beta}_j}{\sqrt{D(\hat{\beta})_{jj}}}$ if the hypothesis $\beta_j = 0$ is true.

Critical Values of Predictor Variables

The critical value of a parameter T statistic for a given level of significance α is the value v_j, such that if the absolute value of the T statistic calculated for a given parameter β_j is greater than v_j, we reject the hypothesis $\beta_j = 0$ at the significance level α.

Prediction Intervals

Suppose we have calculated the parameter estimates $\hat{\beta}$ for our linear regression problem. Suppose further that we have a vector of values, x, for the predictor variables. We may obtain an α level confidence interval for the value, $Y = x'\hat{\beta}$ which is the value of the dependent of the observed variable predicted by our model, according to the formula:

$$x'\hat{\beta} \pm (S\sqrt{x'(X'X)^{-1}x})t(n-p;\alpha/2) \quad (2\text{-}13)$$

where $t(n-p;\alpha/2)$ is the value at $\alpha/2$ of the cumulative distribution function for a T distribution, S is the estimated variance, and X is the regression matrix.

2.2.2 Building Reduced Regression Models

In general, exhaustive search is the only technique guaranteed to find the predictor variable subset with the best evaluation criterion. It is often the ideal technique when the number of possible predictor variables is less than 20; note that this number depends to

some degree on the computational complexity of evaluating a predictor variable subset. The problem with exhaustive search is that it is often a computationally intractable technique for more than 20 possible predictor variables. It is easy to see that there are 2^N-1 possible subsets, excluding the empty set, and exhaustive search must check them all. For regression models with 25 predictor variables, exhaustive search must check 33,554,431 subsets, and this number doubles for each additional predictor variable considered. Clearly, exhaustive search is not always a practical technique, and other selection techniques may have to be considered.

Exhaustive search is not used in this thesis. Instead, the linear procedures such as forward selection, backward elimination, stepwise selection, Maximum R^2 improvement, and Minimum R^2 improvement are employed in feature selection among the 15 cost drivers plus the software size. Three criterions will be applied during the building of reduced models: 1) the R-square value of the reduced model should equal to or greater than 0.95; 2) each of the predictor variable's *p* value is less than 0.05; and 3) the model's *p* value is less than 0.001. The reduced models that meet all the three criteria can only be selected to the optimal reduced models. If there are more than one such optimal models, always pick the one with highest R-square value as the best reduced model.

R-square is called the *coefficient of multiple determination*, it is used to dedicate the goodness of fit of the multiple linear regression models. R-square can be calculated as the following:

$$R^2 = \frac{SSR}{SST} \qquad (2\text{-}14)$$

where SSR is regression sum of squares, and SST is the total sum of squares.

Forward Selection

Unlike exhaustive search, forward selection is always computationally tractable. Even in the worst case, it checks a much smaller number of subsets before finishing. This technique adds predictor variables and never deletes them. The starting subset in forward selection is the empty set. For a regression model with N possible predictor variables, the first step involves evaluating N predictor variable subsets, each consisting of a single predictor variable, and selecting the one with the highest evaluation criterion (R-square value). The next step selects from among N-1 subsets, the next step from N-2 subsets, and so on. Even if all predictor variables are selected, at most N(N+1)/2 subsets are evaluated before the search ends.

The problem with forward selection is that, unlike exhaustive search, it is not guaranteed to find the subset with the highest evaluation criterion. In practice, however, many researchers have reported good results with forward selection [10]. This is not too surprising: it's not hard to show that forward selection will find the subset with the highest evaluation criterion when predictor variables are statistically independent and the observation variable is modeled as a linear combination of predictor variables. Actually, it still holds for an observation variable y and predictor variables x_i such that

$$y = f(\sum_i \beta i x_i) \qquad (2\text{-}15)$$

where f() is any monotonic, continuously differentiable function. While statistical independence of predictor variables may be too much to expect for the regression problem you are trying to improve, it may become more feasible with more study into the predictor variables. You may discover certain preprocessing steps that can be performed to predictor variable data such that the predictor variables become nearly statistically independent.

Backward Selection

Backward selection has computational properties that are similar to forward selection. The starting subset in backward selection includes all possible predictor variables. Predictor variables are deleted one at a time as long as this results in a subset with a higher evaluation criterion. Again, in the worst case, at most N(N+1)/2 subsets must be evaluated before the search ends. Like forward selection, backward selection is not guaranteed to find the subset with the highest evaluation criterion.

Some researchers prefer backward selection to forward selection when the predictor variables are far from statistically independent. In this case, starting the search with all predictor variables included allows the model to take predictor variable interactions into account. Forward selection will not add two predictor variables that together can explain the variations in the observation variable if, individually, the predictor variables are not helpful in explaining the variation. Backward selection, on the other hand, would already include both of these variables and would realize that it is a bad idea to delete either one.

The disadvantage of backward selection is that one's confidence in subset evaluation criterion values tends to be lower than with forward selection. This is especially true when the number of rows in the predictor matrix is close to the number of possible predictor variables. In such a case, there are very few points that the regression model can use in order to determine its parameter values, and the function evaluation criterion will be sensitive to small changes to the predictor matrix data. When the ratio of predictor matrix rows to predictor variables is small, it is usually a better idea to use forward selection than backward selection.

Stepwise Selection

Stepwise selection has been proposed as a technique that combines advantages of forward and backward selection. At any point in the search, a single predictor variable may be added or deleted. Commonly, the starting subset is the empty set. When we think in terms of a predictor variable subset's bit representation, stepwise selection allows one bit in the representation to be flipped at any point in the search. Therefore, since a subset's bit representation has N bits, each subset in the search graph for stepwise selection has N neighbors. If no bit is flipped more than once, at most N^2subsets are evaluated before stepwise selection ends. There are, however, no guarantees that each bit will be flipped at most one time.

No strong theoretical results exist for comparing the effectiveness of stepwise selection against forward or backward selection. Stepwise selection evaluates more subsets than the other two techniques, so in practice it tends to produce better subsets [10]. Of course, the price that stepwise selection pays for finding better subsets is reduced computational speed: usually more subsets must be evaluated before the search ends.

The three stepwise methods described above typically fit only a small number of the possible models when the total number of explanatory variables is large. It is therefore possible to miss the best subset.

Two following recent modifications of the stepwise method are the maximum R^2 improvement and minimum R^2 improvement methods. These two methods move sequentially from level to level where the level is the number of explanatory variables in the model.

Variable Selection Procedure Using Maximum R^2 Improvement

At each level the method attempts to switch variables that are included with variables that have been excluded. The maximum R^2 improvement tries to switch an included variable with an excluded variable that provides the largest increase in R^2. This process continues until no included variable can be switched that would increase R^2.

The resulted model is said to be the best model for the level of explanatory variables. This model, however, is not necessarily the best variable model since not all variable models have been composed. The method then moves to the next level by adding the variable yielding the largest increase in R^2.

Therefore, the models may be conditionally sub-optimal at its level, and the search may be conditionally exhaustive.

Variable Selection Procedure Using Minimum R^2 Improvement

The minimum R^2 improvement method procedure is similar to the maximum R^2 method except at each step within each level the variable chosen for switching is the one providing the smallest increase in R^2. The starting point at each level is the maximum R^2 model from the previous level. The reason for the choice of the model with minimum R^2 at each step within any level is to show the user a large number of reasonable models at each level. Typically for each switch the results of the regression are provided.

Therefore, the models may be conditionally sub-optimal at its level, and the search may be conditionally exhaustive.

Both R^2 improvement procedures are designed to produce a large number of possible regressions at each level. These two methods will tend to yield more possible solutions than the three original stepwise methods. In addition the minimum R^2 method usually produces more solutions than the maximum R^2 method [2].

2.3 Experimental Results and Analysis

2. 3.1. Correlation Analysis between the Input Variables and Output

Correlation analysis measures the relationship between two variables; the degree of association is measured by a correlation coefficient, denoted by ρ. It is sometimes called Pearson's correlation coefficient after its originator and is a measure of linear association. When comparing the correlation between two items, one item is called the "dependent" item and the other the "independent" item. The goal is to see if a change in the independent input (which is our cost drivers) will result in a change in the dependent item. It can be expressed by Equation (2-16):

$$\rho_{x,y} = \frac{Co\,\mathrm{var}\,iance(X,Y)}{\sigma_x \times \sigma_y} \qquad (2\text{-}16)$$

Where $\rho_{x,y}$ is the correlation coefficient between the array X and array Y; Covariance (X,Y) can be calculated by Equation (2-17):

$$Co\,\mathrm{var}\,iance(X,Y) = \frac{1}{N} \times \sum_{i=1}^{N} (x_i - \mu_x)(y_i - \mu_y) \qquad (2\text{-}17)$$

σ_x , σ_y are the standard deviations of variables X and Y, N =63 here, x_i, y_i, μ_x, μ_y are individual element of array X, Y, and the mean of X, and Y. The standard deviations can be calculated by Equation (2-18) and (2-19):

$$\sigma_x = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \mu_x)^2}{N}} \qquad (2\text{-}18)$$

$$\sigma_y = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \mu_y)^2}{N}} \qquad (2\text{-}19)$$

The correlation coefficient can range between ±1.0 (plus or minus one). A coefficient of +1.0, a "perfect positive correlation," means that changes in the independent item will result in an identical change in the dependent item. A coefficient of -1.0, a "perfect negative correlation," means that changes in the independent item will result in an identical change in the dependent item, but the change will be in the opposite direction. A coefficient of zero means there is no relationship between the two items and that a change in the independent item will have no effect in the dependent item.

A low correlation coefficient (e.g., less than ±0.10) suggests that the relationship between two items is weak or non-existent. A high correlation coefficient (i.e., closer to plus or minus one) indicates that the dependent variable will usually change when the independent variable (which is our cost drivers) changes.

The direction of the dependent variable's change depends on the sign of the coefficient. If the coefficient is a positive number, then the dependent variable will move in the same direction as the independent variable; if the coefficient is negative, then the dependent variable will move in the opposite direction of the independent variable.

2.3.2 Applying Multiple Regression Directly on the COCOMO Dataset

First, we need to analyze the Pearson's correlation coefficients of the matrix. From Table 3, we can see that only SIZE, MODP and DATA are significant to Effort at the level of 0.05, other explanatory variables seem to have very weak linear relationships with the dependent actual effort. From Table 4, we can further conclude that this linear model is very reluctant since the R-Square is low at 0.6384 and C (p) = 17.00. We can assume that a non-linear model between the dependent variable effort and other 16 explanatory variables should be more reasonable.

Table 2-1. Pearson's Correlation Matrix between the 16 Explanatory Variables and the Dependent Variable Effort

	RELY	*DATA*	*CPLX*	*TIME*	*STOR*	*VIRT*	*TURN*	*ACAP*	*AEXP*	*PCAP*	*VEXP*	*LEXP*	*MODP*	*TOOL*	*SCED*	*SIZE*	*Effort*
RELY	1																
DATA	-0.031	1															
CPLX	**0.559**	-0.328	1														
TIME	**0.703**	-0.097	**0.487**	1													
STOR	**0.654**	-0.068	**0.518**	**0.678**	1												
VIRT	0.303	-0.215	0.3131	**0.445**	**0.416**	1											
TURN	-0.015	0.163	-0.021	0.012	0.187	0.206	1										
ACAP	**-0.444**	0.152	**-0.460**	-0.323	-0.192	-0.156	0.130	1									
AEXP	-0.235	0.089	-0.120	-0.142	-0.209	0.061	0.015	0.377	1								
PCAP	-0.327	0.295	**-0.451**	-0.258	-0.179	-0.163	0.120	**0.668**	0.098	1							
VEXP	0.138	-0.148	0.303	0.297	0.155	**0.698**	-0.030	-0.277	0.231	-0.202	1						
LEXP	0.342	-0.250	**0.480**	**0.442**	0.372	**0.692**	0.038	-0.378	0.043	-0.307	**0.797**	1					
MODP	-0.216	0.129	-0.128	-0.080	0.108	0.136	**0.499**	0.384	-0.052	**0.530**	0.072	0.099	1				
TOOL	-0.041	-0.263	0.265	0.148	0.221	**0.519**	0.343	0.028	-0.146	0.031	**0.429**	**0.437**	**0.471**	1			
SCED	0.117	-0.041	0.135	0.120	-0.060	0.254	0.094	0.091	0.352	-0.085	0.176	0.305	-0.020	0.141	1		
SIZE	0.041	0.367	-0.201	-0.055	-0.035	-0.189	0.083	-0.138	-0.111	0.158	-0.079	-0.069	0.060	-0.168	-0.102	1	
Effort	0.207	**0.445**	0.010	0.153	0.105	0.019	0.206	-0.148	-0.036	0.157	0.068	0.088	0.270	0.002	0.021	**0.657**	1

Table 2-2. Full Multiple Regression Model of the Dependent Variable Effort and the 16 Explanatory Variables

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	16	130587330	8161708	5.00	<.0001
Error	46	75122690	1633102		
Corrected Total	62	205710019			

Variable	Parameter Estimate	Standard Error	Type II SS	F Value	Pr > F
Intercept	-12385	5814.20186	7409592	4.54	0.0385
SOFT_SIZE	6.09620	1.14781	46066903	**28.21**	**<.0001**
RELY	1669.04182	1536.83438	1926165	1.18	0.2831
DATA	6483.60033	2710.96352	9341109	**5.72**	**0.0209**
CPLX	1194.52192	1262.02315	1463076	0.90	0.3488
TIME	1541.94229	1632.23866	1457412	0.89	0.3498
STOR	-1891.12897	1639.71348	2172299	1.33	0.2547
VIRT	1478.71007	2465.60026	587400	0.36	0.5516
TURN	-814.27555	2679.24966	150845	0.09	0.7626
ACAP	-2332.65400	2033.51092	2148923	1.32	0.2573
AEXP	2324.33364	2028.79782	2143545	1.31	0.2579
PCAP	251.73075	1633.80136	38769	0.02	0.8782
VEXP	-1596.37034	3989.74282	261451	0.16	0.6909
LEXP	-1667.78613	6817.87138	97723	0.06	0.8078
MODP	4972.31237	2053.28572	9577041	**5.86**	**0.0195**
TOOL	759.12041	3050.92862	101105	0.06	0.8046
SCED	-21.50974	2820.13779	95.00426	0.00	0.9939

*R-Square = 0.6384 C(p) = 17.0000

Such model may be used but the linear relationships between the dependent variable and the 16 explanatory variables are very poor, and it cannot meet the effort prediction objective expected when dealing with our COCOMO dataset. The variation of the model is only 63.84% since the R-Square is low at 0.6384. Thus we can assume that a non-linear model between the dependent variable Effort and other 16 explanatory variables by transferring the software size (size) to logarithm to size and logarithm to the dependent variable effort in Section 2.3.3.

2.3.3. Applying Multiple Regression Indirectly on the COCOMO Dataset

Although the relation between the dependent variable effort and the 16 explanatory variables especially the software size is regarded as nonlinear, the technique of linear regression can still be applied if the model can be written as a linear function of variables which themselves are nonlinear. Simple linear models involving variables such as log X, log Y, 1/Y, 1/X, X^2 and Y^2 are examples. There are a number of nonlinear functions relating effort to software size and other independent variables, which may be converted to a linear function by suitable transformations on Y (for our case, effort) and /or X (for our case, software size). This class of models is called intrinsically linear [2]. Here we choose the format of: $\log Y = \beta_0 + \beta_1 \log X$. Other 15 explanatory variables do not transform to logarithm form. After being transferred to logarithm size and logarithm effort, calculate the Pearson's correlation coefficients matrix for the modified COCOMO dataset as Table 2-3.

Table 2-3. Pearson's Correlation Matrix between the 15 Explanatory Variables plus ln(size) and the Dependent Variable ln(Effort)

	RELY	*DATA*	*CPLX*	*TIME*	*STOR*	*VIRT*	*TURN*	*ACAP*	*AEXP*	*PCAP*	*VEXP*	*LEXP*	*MODP*	*TOOL*	*SCED*	*lnsize*	*lneff*
RELY	1																
DATA	-0.031	1															
CPLX	**0.559**	-0.328	1														
TIME	**0.703**	-0.097	**0.488**	1													
STOR	**0.654**	-0.068	**0.518**	**0.678**	1												
VIRT	0.303	-0.215	0.313	**0.445**	**0.416**	1											
TURN	-0.015	0.163	-0.021	0.012	0.187	0.206	1										
ACAP	**-0.444**	0.152	**-0.460**	-0.323	-0.192	-0.156	0.130	1									
AEXP	-0.235	0.089	-0.120	-0.142	-0.209	0.061	0.015	0.377	1								
PCAP	-0.327	0.295	**-0.451**	-0.258	-0.179	-0.163	0.120	**0.668**	0.098	1							
VEXP	0.138	-0.148	0.303	0.297	0.155	**0.698**	-0.030	-0.277	0.231	-0.202	1						
LEXP	0.342	-0.250	**0.480**	**0.442**	0.372	**0.692**	0.038	-0.378	0.043	-0.307	**0.797**	1					
MODP	-0.217	0.129	-0.128	-0.080	0.108	0.136	**0.499**	0.384	-0.052	**0.530**	0.072	0.099	1				
TOOL	-0.041	-0.263	0.265	0.148	0.221	**0.519**	0.343	0.028	-0.146	0.031	**0.429**	**0.437**	**0.471**	1			
SCED	0.117	-0.041	0.135	0.120	-0.060	0.254	0.094	0.091	0.352	-0.085	0.176	0.305	-0.020	0.141	1		
lnsize	0.057	**0.546**	-0.301	-0.009	-0.062	-0.182	0.255	-0.147	-0.204	0.129	-0.133	-0.149	0.061	-0.228	-0.167	1	
lneff	0.310	**0.514**	-0.058	0.314	0.228	0.163	0.344	-0.066	-0.090	0.207	0.142	0.152	0.222	0.022	0.051	**0.847**	1

It can be seen from Table 2-3 that the relationships between the 15 explanatory variables have not been changed, but the ln(size) has become more significant to ln(effort) than that of size to effort. The correlation coefficient changes from 0.658 between effort vs. size to 0.847 between ln(effort) vs. ln(size). Furthermore, the correlation coefficient between effort vs. DATA has grown to 0.514 between ln(effort) vs. DATA. From this point of view, it can be concluded that the logarithm transformation of software size and effort is suitable because the relationship between the significant explanatory variables and the dependent variable effort have become more significant.

The next step is to build the full and reduced regression model between the effort and size indirectly, using logarithm transferring methods as preprocessing techniques. The full multiple regression model is calculated and shown in Table 2-4, where some advantages

are found by comparing the direct full multiple regression model in Table 2-2 with that of Table 2-4.

Table 2-4. Full Multiple Regression Model of the Dependent Variable ln(Effort) and the 15 Explanatory Variables plus ln(size)

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	16	197.81292	12.36331	64.88	<.0001
Error	46	8.76514	0.19055		
Corrected Total	62	206.57806			

Variable	Parameter Estimate	Standard Error	Type II SS	F Value	Pr > F
Intercept	-17.01783	1.98405	14.01859	73.57	<.0001
ln_size	**1.16263**	**0.05691**	**79.51759**	**417.31**	**<.0001**
RELY	**1.53326**	**0.52533**	**1.62321**	**8.52**	**0.0054**
DATA	1.43248	1.00093	0.39028	2.05	0.1591
CPLX	**0.96500**	**0.43883**	**0.92145**	**4.84**	**0.0329**
TIME	**1.62272**	**0.55987**	**1.60072**	**8.40**	**0.0057**
STOR	0.25883	0.56066	0.04061	0.21	0.6465
VIRT	1.06389	0.83686	0.30796	1.62	0.2100
TURN	-0.01419	0.97654	0.00004021	0.00	0.9885
ACAP	**1.77751**	**0.68464**	**1.28440**	**6.74**	**0.0126**
AEXP	0.75896	0.70595	0.22024	1.16	0.2879
PCAP	**1.49816**	**0.54902**	**1.41890**	**7.45**	**0.0090**
VEXP	2.27808	1.36557	0.53029	2.78	0.1021
LEXP	0.54306	2.32562	0.01039	0.05	0.8164
MODP	0.51959	0.70262	0.10420	0.55	0.4634
TOOL	1.56479	1.05121	0.42221	2.22	0.1434
SCED	**2.06952**	**0.96506**	**0.87626**	**4.60**	**0.0373**

*R-Square = 0.9576 and C(p) = 17.0000

From Table 2-4, it is clearly seen that the p-values of the explanatory variables ln_size, RELY, CPLX, TIME, ACAP, PCAP and SCED are significant at level of 0.05, thus they can make significant contributions to the dependent variable ln_effort, most of which could not be identified from the correlation matrix analysis from Table 2-3. Furthermore, the R-Square equals 0.9576, which means that 95.76% of the total variation can be covered in the model, this, again confirms the linear relationship between ln_effort and the explanatory variables.

Based on the above analysis, reduced model relating ln_effort with ln_size, RELY, CPLX, TIME, ACAP, PCAP and SCED could be calculated, as Table 2-5. This model is

only the primary reduced multiple regression model, more effort should be done for the comparison of a series of reduced models in later sections, in order to find the potential outliers of the dataset and the optimal reduced regression model for more efficient prediction of effort of the COCOMO dataset.

Table 2-5. Reduced Multiple Regression Model of the Dependent Variable ln(Effort) and the 6 Explanatory Variables plus ln(size)

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	7	187.44488	26.77784	76.98	<.0001
Error	55	19.13319	0.34788		
Corrected Total	62	206.57806			

Root MSE	0.58981	R-Square	0.9074
Dependent Mean	4.82848	Adj R-Sq	0.8956
Coeff Var	12.21524		

Parameter Estimates

Variable	Label	DF	Parameter Estimate	Standard Error	Standardized t Value	Pr > |t|
Intercept	Intercept	1	-11.59607	1.44128	-8.05	<.0001
ln_size	ln_size	1	1.16286	0.06148	18.91	<.0001
RELY	RELY	1	0.83482	0.60682	1.38	0.1745
CPLX	CPLX	1	1.53641	0.53590	2.87	0.0059
TIME	TIME	1	2.82168	0.66059	4.27	<.0001
ACAP	ACAP	1	1.57690	0.77511	2.03	0.0467
PCAP	PCAP	1	2.01758	0.64565	3.12	0.0028
SCED	SCED	1	3.49711	1.03829	3.37	0.0014

From Table 2-5, it can be clearly seen that only the p-value for the explanatory variable RELY is not significant, which is not consistent with the situation in the full model of regression relating effort with other 16 explanatory variables. The R-Square equals 0.9074, compared with the R-Square equals 0.9576 in the full regression model, it is also good enough for the variation covered.

2.3.4. Detection of Potential Outliers and the Influential Effects on the Reduced Regression Model for the Transformed COCOMO Dataset

The fitting of a multiple linear regression model to a sample of observations is not complete without an assessment of the sensitivity of the results to the observed data. Some observation can have large influence on the magnitude of the estimated coefficients, they may or may not be outliers. Some techniques have been developed to help identify outliers and influential observations.

A useful way of judging outliers is to determine whether the extreme point is within a predictable range, given a linear relationship that would be fitted without this extreme point. The standardized residuals, studentized residuals, Cook's D, H-hat, the DFFITS family and COVRATIO statistic are measured, if they are higher for some observations than those of the most observations, it can be concluded that those observations are potential outliers, but their influence may or may not be significant to the model, it needs to delete the potential outliers and investigate their significance on the models, either one by one or delete all of the potential outliers at once. Table 2-6 shows the influence diagnostics for the reduced regression model for the transformed COCOMO dataset.

Table 2-6. Influence Diagnostics for Reduced Model on the COCOMO Dataset

Obs	Dep Var ln_effort	Predicted Value	Std Error Mean Predict	Std Error Residual	Std Error Residual	Student Residual	-2-1 0 1 2	Cook's D
1	7.6207	6.4071	0.2000	1.2136	0.555	2.187	****	0.078
2	7.3778	6.9631	0.1785	0.4147	0.562	0.738	*	0.007
3	5.4931	5.6328	0.1667	-0.1397	0.566	-0.247		0.001
4	5.4806	6.0668	0.2438	-0.5862	0.537	-1.092	**	0.031
5	3.4965	3.5299	0.1495	-0.0334	0.571	-0.0586		0.000
6	3.7612	3.4341	0.3108	0.3271	0.501	0.653	*	0.020
7	2.0794	2.7258	0.1580	-0.6464	0.568	-1.137	**	0.013
8	6.9801	6.5538	0.3663	0.4263	0.462	0.922	*	0.067
9	6.0474	5.5730	0.1511	0.4744	0.570	0.832	*	0.006
10	5.7714	5.2062	0.2334	0.5653	0.542	1.044	**	0.025
11	5.3845	5.3206	0.2338	0.0639	0.541	0.118		0.000
12	5.3033	5.5605	0.1380	-0.2572	0.573	-0.449		0.001
13	4.3694	4.2655	0.1581	0.1039	0.568	0.183		0.000
14	4.2905	3.9147	0.2929	0.3758	0.512	0.734	*	0.022
15	4.1109	4.2135	0.2500	-0.1027	0.534	-0.192		0.001
16	3.6889	3.9294	0.2039	-0.2405	0.553	-0.435		0.003
17	2.1972	3.3161	0.2193	-1.1189	0.548	-2.043	****	0.084
18	9.3414	8.6574	0.1940	0.6840	0.557	1.228	**	0.023
19	8.7948	8.8620	0.2364	-0.0672	0.540	-0.124		0.000
20	8.7641	8.4064	0.3010	0.3577	0.507	0.705	*	0.022
21	7.8059	7.2974	0.1776	0.5085	0.562	0.904	*	0.010
22	6.5848	7.4243	0.2431	-0.8395	0.537	-1.562	***	0.062
23	6.2897	6.3917	0.2357	-0.1020	0.541	-0.189		0.001
24	6.1159	5.7598	0.1420	0.3561	0.572	0.622	*	0.003
25	6.2596	5.7044	0.1298	0.5552	0.575	0.965	*	0.006
26	5.9584	4.8836	0.1563	1.0748	0.569	1.890	***	0.034
27	4.4773	4.9171	0.1653	-0.4398	0.566	-0.777	*	0.006
28	4.5850	4.0644	0.1393	0.5205	0.573	0.908	*	0.006
29	1.9879	2.1525	0.2540	-0.1647	0.532	-0.309		0.003
30	1.7750	1.9045	0.2477	-0.1295	0.535	-0.242		0.002
31	6.9689	6.6728	0.2576	0.2960	0.531	0.558	*	0.009
32	6.5539	6.8383	0.2338	-0.2844	0.542	-0.525	*	0.006
33	6.4052	6.6809	0.2141	-0.2756	0.550	-0.502	*	0.005
34	5.4381	5.4334	0.2286	0.004635	0.544	0.00852		0.000
35	4.4067	4.0926	0.2388	0.3141	0.539	0.582	*	0.008
36	4.0073	4.2894	0.1661	-0.2820	0.566	-0.498		0.003
37	3.8501	4.3853	0.2059	-0.5351	0.553	-0.968	*	0.016
38	2.4849	3.6107	0.2117	-1.1258	0.551	-2.045	****	0.077
39	2.0794	1.9780	0.1900	0.1014	0.558	0.182		0.000
40	2.0794	2.5491	0.2718	-0.4697	0.523	-0.897	*	0.027
41	1.7918	1.9223	0.2338	-0.1305	0.541	-0.241		0.001
42	3.8067	4.9132	0.1203	-1.1065	0.577	-1.916	***	0.020
43	4.4188	4.6147	0.1177	-0.1958	0.578	-0.339		0.001
44	4.4659	4.9831	0.1150	-0.5172	0.578	-0.894	*	0.004
45	4.6634	4.9701	0.1183	-0.3066	0.578	-0.531	*	0.001
46	4.8363	5.5424	0.1795	-0.7062	0.562	-1.257	**	0.020
47	3.5835	3.6641	0.2546	-0.0806	0.532	-0.151		0.001
48	7.1483	8.2801	0.2455	-1.1318	0.536	-2.110	****	0.117
49	5.0499	4.6409	0.2205	0.4089	0.547	0.748	*	0.011
50	5.1705	4.8526	0.1670	0.3179	0.566	0.562	*	0.003
51	4.8040	4.5527	0.2489	0.2513	0.535	0.470		0.006
52	3.7136	2.8046	0.1571	0.9090	0.569	1.599	***	0.024
53	2.6391	2.2856	0.2089	0.3534	0.552	0.641	*	0.007
54	2.9957	2.4113	0.1632	0.5844	0.567	1.031	**	0.011
55	2.8904	2.3899	0.3082	0.5005	0.503	0.995	*	0.046
56	6.8648	6.0127	0.1528	0.8522	0.570	1.496	**	0.020
57	5.4681	5.3283	0.2043	0.1398	0.553	0.253		0.001
58	4.8675	6.0261	0.2917	-1.1586	0.513	-2.260	****	0.207
59	4.2485	4.7343	0.1246	-0.4859	0.576	-0.843	*	0.004
60	4.0431	3.5734	0.1324	0.4696	0.575	0.817	*	0.004
61	3.9120	4.2906	0.1231	-0.3785	0.577	-0.656	*	0.002
62	3.6376	2.9753	0.2014	0.6623	0.554	1.195	**	0.024
63	2.7081	2.8567	0.1783	-0.1487	0.562	-0.264		0.001

Table 2-6. Influence Diagnostics for Reduced Model on the COCOMO Dataset (Cont.)

Obs	RStudent	Hat Diag H	Cov Ratio	DFFITS
1	2.2681	0.1150	0.6319	0.8177
2	0.7346	0.0916	1.1774	0.2333
3	-0.2448	0.0799	1.2475	-0.0721
4	-1.0935	0.1709	1.1723	-0.4964
5	-0.0581	0.0642	1.2370	-0.0152
6	0.6491	0.2777	1.5068	0.4025
7	-1.1406	0.0718	1.0313	-0.3171
8	0.9207	0.3856	1.6642	0.7295
9	0.8297	0.0657	1.1200	0.2199
10	1.0445	0.1566	1.1701	0.4501
11	0.1169	0.1571	1.3713	0.0505
12	-0.4453	0.0548	1.1898	-0.1072
13	0.1813	0.0719	1.2418	0.0505
14	0.7309	0.2466	1.4207	0.4182
15	-0.1905	0.1796	1.4041	-0.0891
16	-0.4313	0.1195	1.2796	-0.1589
17	-2.1064	0.1382	0.7145	-0.8435
18	1.2338	0.1082	1.0396	0.4297
19	-0.1233	0.1607	1.3767	-0.0539
20	0.7019	0.2604	1.4561	0.4164
21	0.9026	0.0907	1.1298	0.2850
22	-1.5835	0.1699	0.9704	-0.7164
23	-0.1871	0.1597	1.3711	-0.0815
24	0.6186	0.0580	1.1619	0.1535
25	0.9644	0.0484	1.0617	0.2176
26	1.9365	0.0702	0.7279	0.5321
27	-0.7739	0.0785	1.1507	-0.2260
28	0.9068	0.0558	1.0869	0.2204
29	-0.3068	0.1854	1.4021	-0.1464
30	-0.2399	0.1764	1.3942	-0.1110
31	0.5544	0.1907	1.3675	0.2691
32	-0.5217	0.1571	1.3198	-0.2252
33	-0.4981	0.1318	1.2858	-0.1940
34	0.008447	0.1502	1.3628	0.0036
35	0.5788	0.1639	1.3182	0.2562
36	-0.4949	0.0793	1.2131	-0.1452
37	-0.9676	0.1219	1.1494	-0.3605
38	-2.1080	0.1288	0.7061	-0.8107
39	0.1800	0.1038	1.2861	0.0613
40	-0.8956	0.2123	1.3066	-0.4650
41	-0.2390	0.1572	1.3625	-0.1032
42	-1.9656	0.0416	0.6952	-0.4095
43	-0.3361	0.0398	1.1862	-0.0684
44	-0.8925	0.0380	1.0708	-0.1774
45	-0.5272	0.0402	1.1581	-0.1079
46	-1.2637	0.0926	1.0109	-0.4037
47	-0.1501	0.1864	1.4186	-0.0719
48	-2.1813	0.1733	0.7128	-0.9986
49	0.7445	0.1398	1.2407	0.3001
50	0.5585	0.0801	1.2023	0.1648
51	0.4666	0.1781	1.3644	0.2172
52	1.6224	0.0710	0.8518	0.4484
53	0.6373	0.1254	1.2471	0.2413
54	1.0317	0.0765	1.0728	0.2970
55	0.9951	0.2730	1.3774	0.6098
56	1.5133	0.0671	0.8905	0.4059
57	0.2505	0.1200	1.3039	0.0925
58	-2.3513	0.2446	0.7029	-1.3380
59	-0.8405	0.0447	1.0925	-0.1817
60	0.8146	0.0504	1.1061	0.1877
61	-0.6528	0.0436	1.1371	-0.1394
62	1.1994	0.1166	1.0622	0.4357
63	-0.2622	0.0914	1.2617	-0.0832

Table 2-6. Influence Diagnostics for Reduced Model on the COCOMO Dataset (Cont.)

				DFBETAS				
Obs	Intercept	ln_size	RELY	CPLX	TIME	ACAP	PCAP	SCED
1	-0.1007	0.2558	0.1042	-0.2653	0.0184	0.3958	-0.0977	-0.0233
2	0.0205	0.1650	-0.0327	-0.0040	-0.0042	0.0866	-0.0722	-0.0424
3	-0.0355	-0.0207	-0.0183	0.0345	0.0183	-0.0033	0.0298	0.0138
4	0.0953	-0.0019	0.1011	0.0902	-0.1015	0.0568	-0.3009	-0.0438
5	-0.0070	-0.0002	0.0017	0.0005	0.0014	-0.0082	0.0101	0.0064
6	-0.0613	-0.0821	-0.0147	0.0269	0.0482	0.1984	0.0728	-0.1041
7	-0.1342	0.1143	0.1768	-0.0101	-0.0644	0.0081	-0.0095	0.0936
8	-0.1672	-0.0699	-0.3875	-0.0008	0.6120	-0.3090	0.2957	0.0896
9	-0.0122	0.0584	-0.0537	0.0984	0.1021	0.0664	-0.0443	-0.0961
10	0.0254	0.0242	0.3602	0.0746	-0.2737	0.1336	-0.0634	-0.1422
11	0.0024	0.0040	0.0401	0.0089	-0.0307	0.0154	-0.0074	-0.0158
12	0.0356	-0.0464	-0.0200	-0.0670	0.0454	-0.0270	0.0165	-0.0175
13	0.0225	0.0043	0.0069	0.0139	-0.0170	-0.0026	-0.0189	-0.0168
14	-0.2284	0.0240	-0.1231	0.2493	0.0511	0.0816	-0.0740	0.1963
15	0.0424	0.0339	-0.0419	0.0156	0.0086	0.0092	-0.0107	-0.0542
16	-0.0202	0.0656	-0.1027	0.0150	0.0123	-0.0266	0.0087	0.0662
17	-0.1438	0.4488	-0.5289	0.1259	0.0596	-0.0888	0.0149	0.3355
18	-0.2294	0.3045	-0.1025	0.1121	0.1718	0.0042	0.0725	0.0993
19	-0.0010	-0.0315	-0.0059	0.0007	0.0073	0.0199	-0.0156	-0.0002
20	-0.1265	0.1244	0.1876	0.1059	-0.1990	-0.1432	0.2226	0.0895
21	-0.0293	0.2151	-0.0604	0.1593	-0.0416	0.0068	0.0410	-0.0421
22	0.3821	-0.2181	0.0316	0.2015	-0.1886	0.0460	0.0811	-0.5547
23	0.0345	-0.0168	-0.0227	0.0213	0.0212	0.0077	0.0075	-0.0661
24	0.0374	0.0577	-0.0378	-0.0422	0.0400	0.0440	-0.0420	-0.0421
25	-0.0852	0.1025	0.0175	0.1395	-0.0537	0.0585	-0.0363	0.0363
26	0.1446	-0.0086	-0.0555	-0.3604	0.0753	-0.1275	-0.1425	0.1637
27	0.0847	0.0582	0.0159	0.0242	-0.1506	-0.0488	-0.0199	-0.0076
28	0.0483	-0.0164	0.0691	0.1013	-0.0870	0.0481	-0.0299	-0.1029
29	0.0364	0.0456	0.0016	0.0289	0.0130	-0.0379	0.0484	-0.0872
30	0.0183	0.0495	0.0059	0.0320	0.0094	0.0060	0.0156	-0.0750
31	0.0165	-0.0101	0.0917	-0.1436	0.1306	0.0427	-0.0588	-0.0736
32	-0.0672	-0.0646	0.0584	0.0496	-0.0008	0.1469	-0.0770	-0.0158
33	0.0322	-0.0304	-0.0393	-0.0121	-0.0927	-0.0657	0.0344	0.0734
34	-0.0019	-0.0001	0.0016	-0.0009	-0.0012	0.0001	0.0003	0.0027
35	-0.0140	0.0524	-0.1783	0.1893	0.0521	0.0519	0.0095	-0.0582
36	0.0095	0.0245	-0.0199	0.0318	0.0053	-0.0650	-0.0093	0.0164
37	-0.2258	0.0375	-0.0040	0.2771	-0.0193	0.0692	0.1101	0.0438
38	-0.3082	0.3022	0.0063	-0.0142	0.2379	0.6532	-0.5238	0.0469
39	0.0460	-0.0288	0.0037	-0.0111	-0.0199	-0.0223	-0.0144	-0.0143
40	-0.0511	0.2214	-0.0103	-0.1376	0.1283	0.2331	-0.3479	0.0551
41	-0.0510	0.0190	0.0012	0.0182	0.0057	-0.0538	0.0824	0.0364
42	-0.0831	-0.1258	0.2007	-0.1165	0.0634	0.1123	-0.0278	-0.0374
43	-0.0142	0.0084	-0.0104	-0.0023	0.0349	0.0350	-0.0340	-0.0057
44	0.0197	-0.0604	0.0964	-0.0755	-0.0296	-0.0593	0.0129	0.0132
45	0.0083	-0.0435	0.0377	-0.0529	0.0187	-0.0383	0.0093	0.0059
46	0.0109	-0.2549	0.1067	-0.1566	0.0470	-0.2552	0.2493	0.0373
47	-0.0163	-0.0152	0.0513	-0.0347	-0.0032	0.0148	0.0134	-0.0030
48	0.3457	-0.7442	0.0569	-0.2104	0.0309	-0.5956	0.1507	0.0167
49	0.1963	0.0117	0.0427	-0.1720	-0.0545	-0.0715	-0.1291	-0.0326
50	0.0554	-0.0402	0.0140	-0.0976	0.0920	0.0011	-0.0398	-0.0573
51	-0.0592	-0.0488	-0.0002	0.0407	-0.0151	-0.0125	0.1607	-0.0012
52	0.2653	-0.2671	0.0379	-0.2706	-0.0044	-0.0254	-0.0160	-0.1363
53	0.0926	-0.0938	-0.1245	-0.0363	0.0572	-0.1178	-0.0334	0.0477
54	0.1394	-0.2020	0.1245	-0.0955	-0.0988	0.0155	0.0148	-0.1105
55	0.2636	-0.4061	0.0448	-0.3958	0.0239	-0.4172	0.3336	-0.0107
56	-0.2125	0.0504	-0.1173	0.1795	0.1785	-0.0801	0.1953	0.0792
57	-0.0676	0.0098	-0.0140	0.0177	-0.0010	0.0022	0.0139	0.0801
58	-0.0462	0.0719	0.1555	0.2477	-1.0255	-0.0326	0.2845	0.3576
59	-0.0064	-0.0349	-0.0055	-0.0943	0.0392	-0.0910	0.0122	0.0867
60	-0.0127	-0.0643	0.0625	0.0602	-0.0818	-0.0016	0.0035	0.0215
61	-0.0633	-0.0222	-0.0116	-0.0467	0.0731	-0.0167	0.0388	0.0537
62	0.1685	-0.0035	-0.2670	0.1749	0.0861	0.0025	-0.1638	-0.1199
63	-0.0510	0.0389	-0.0021	0.0065	0.0289	0.0574	-0.0243	0.0127

From Table 2-6, it can be seen that the standardized residuals, studentized residuals, Cook's D, H-hat, the DFFITS family and COVRATIO statistic for observations 1, 17, 38, 48, 58 are much higher that those of other observations as highlighted in red in Table 2-6.

Therefore, it can be concluded that observations 1, 17, 38, 48, 58 are potential outliers for the reduced regression model for the transformed COCOMO dataset.

To conveniently compare the reduced models, the model with the presence of outliers is listed as Equation (2-20):

$$\begin{aligned} \text{Ln_effort} = &-11.596 + 1.163*\text{ln_size} + \mathbf{0.835}*\text{RELY} + \mathbf{1.536}*\text{CPLX} \\ &+ 2.822*\text{TIME} + \mathbf{1.577}*\text{ACAP} + 2.018*\text{PCAP} \\ &+ 3.497*\text{SCED} \end{aligned} \quad (2\text{-}20)$$

This model has R-Square equals 0.9074; only the coefficient of RELY is not significant at the level of 0.05.

After deleting the No.1 observation, the reduced model becomes:

$$\begin{aligned} \text{Ln_effort} = &-11.456 + 1.148*\text{ln_size} + \mathbf{0.774}*\text{RELY} + \mathbf{1.674}*\text{CPLX} \\ &+ 2.810*\text{TIME} + \mathbf{1.281}*\text{ACAP} + 2.078*\text{PCAP} \\ &+ 3.521*\text{SCED} \end{aligned} \quad (2\text{-}21)$$

This model has R-Square equals 0.9121; besides the coefficient of RELY is not significant at the level of 0.05; it also causes the coefficient of ACAP to be not significant at the level of 0.05; deleting No.1 observation will result in a worse regression, therefore No.1 observation is not an outlier but an influential point.

After deleting the No.17 observation, the reduced model becomes:

$$\begin{aligned} \text{Ln_effort} = &-11.395 + 1.136*\text{ln_size} + \mathbf{1.146}*\text{RELY} + \mathbf{1.471}*\text{CPLX} \\ &+2.784*\text{TIME} + \mathbf{1.644}*\text{ACAP} + 2.008*\text{PCAP} \\ &+ 3.159*\text{SCED} \end{aligned} \quad (2\text{-}22)$$

This model has R-Square = 0.9114; deleting No.17 observation will result in better regression: the coefficient of RELY is significant at the level of 0.10, other coefficients

are all kept significant at level 0.05, the coefficient of RELY has greatly increased from 0.83482 to 1.14619, other coefficients do not change significantly, so No.17 observation is an outlier.

After deleting the No.38 observation, the reduced model becomes:

$$\begin{aligned} \text{Ln_effort} = &-11.165 + 1.145 * \text{ln_size} + \mathbf{0.831} * \text{RELY} + \mathbf{1.544} * \text{CPLX} \\ &+ 2.669 * \text{TIME} + \mathbf{1.086} * \text{ACAP} + 2.346 * \text{PCAC} \\ &+ 3.450 * \text{SCED} \end{aligned} \quad (2\text{-}23)$$

This model has R-Square equals 0.9120; besides the coefficient of RELY is not significant at the level of 0.05 (status not changed); it also causes the coefficient of ACAP to be not significant at the level of 0.05 (status changed); deleting No.38 observation will result in a worse regression, therefore No.38 observation is not an outlier but an influential point.

After deleting the No.48 observation, the reduced model becomes:

$$\begin{aligned} \text{Ln_effort} = &-12.078 + 1.207 * \text{ln_size} + \mathbf{0.801} * \text{RELY} + \mathbf{1.646} * \text{CPLX} \\ &+ 2.802 * \text{TIME} + \mathbf{2.024} * \text{ACAP} + 1.923 * \text{PCAP} \\ &+ 3.480 * \text{SCED} \end{aligned} \quad (2\text{-}24)$$

This model has R-Square equals 0.9126; besides the coefficient of RELY is not significant at the level of 0.05 (status not changed); other coefficients do not change significantly; deleting No.48 observation has little impact on the original regression, therefore, No.48 observation is not an outlier but an influential point.

After deleting the No.58 observation, the reduced model becomes:

$$\begin{aligned} \text{Ln_effort} = &-11.532 + 1.159 * \text{ln_size} + \mathbf{0.744} * \text{RELY} + \mathbf{1.409} * \text{CPLX} \\ &+ 3.473 * \text{TIME} + \mathbf{1.601} * \text{ACAP} + 1.841 * \text{PCAC} \end{aligned}$$

$$+ 3.140 * SCED \qquad (2\text{-}25)$$

This model has R-Square equals 0.9160; besides the coefficient of RELY is not significant at the level of 0.05 (status not changed); other coefficients do not change significantly; deleting No.58 observation has little impact on the original regression, therefore, No.58 observation is not an outlier but an influential point.

After deleting all No.1, 17, 38, 48, 58 potential outliers, the reduced model becomes:

$$\text{Ln_effort} = -11.072 + \mathbf{1.126}* \text{ln_size} + \mathbf{0.999} * \text{RELY} + 1.546 * \text{CPLX}$$
$$+ 3.294 * \text{TIME} + \mathbf{1.207} * \text{ACAP} + 2.138* \text{PCAC}$$
$$+ 2.708 * \text{SCED} \qquad (2\text{-}26)$$

This model has R-Square equals **0.9365**, it has been enhanced; the coefficient of RELY is significant at the level of 0.10 (status changed); other coefficients have also changed significantly; deleting No.1, 17, 38, 48, 58 observations will result in positive improvement on the original regression, although not all No. 1, 17, 38, 48, 58 observations are outliers, their influential impacts on the original regression model are significantly negative.

2.3.5. Detection of Potential Outliers and the Influential Effects on the Full Regression Model for the Transformed COCOMO Dataset

As listed in Section 2.3.2, the full regression model relating ln(Effort) to all 16 explanatory variables for the transformed COCOMO dataset is shown as following:

$$\text{Ln_effort} = -17.018 + \mathbf{1.163}*\text{ln_size} + \mathbf{1.533}*\text{RELY} + 1.433*\text{DATA}$$
$$+ \mathbf{0.965}*\text{CPLX} + \mathbf{1.623}*\text{TIME} + 0.259*\text{STOR} + 1.064*\text{VIRT}$$

-0.0142*TURN+**1.778***ACAP+0.759*AEXP+**1.498***PCAP

+2.278*VEXP+0.543*LEXP+0.520*MODP+1.565*TOOL

+ **2.070***SCED (2-27)

This full regression model has R-Square = 0.9576; as it can be seen that there are only seven coefficients: ln_size, RELY, CPLX, TIME, ACAP, PCAP, SCED are significant to ln_effort at level of 0.05.

As the previously mentioned methods, that is, the standardized residuals, studentized residuals, Cook's D, H-hat, the DFFITS family and COVRATIO statistic, if they are higher for some observations than those of the most observations, it is concluded that those observations are potential outliers, but their influence may or may not be significant to the model, it needs to delete the potential outliers and investigate their significance on the models, either one by one or delete all of the potential outliers at once. Just skip the influence diagnostics procedure for the full regression model for the transformed COCOMO dataset.

It can be detected that observations No.17, No.55, and No.56 are potential outliers according to the techniques mentioned before.

After deleting observation No.17, the model becomes Equation (2-28):

Ln_effort = -16.449+ **1.129** *ln_size + 1.749*RELY + **1.666***DATA

+0.850*CPLX+1.463*TIME + **0.679***STOR + 0.780*VIRT

-0.417*TURN+1.832*ACAP+0.493*AEXP+ 1.372*PCAP

+**2.935***VEXP-**0.430***LEXP+0.740*MODP+1.440*TOOL

+ 2.208*SCED (2-28)

This model has RMSE = 0.39965, R-Square = 0.9640; Compared to Equation (2-27), where R-Square = 0.9576, and seven coefficients: ln_size, RELY, CPLX, TIME, ACAP, PCAP, SCED are significant to ln_effort at level of 0.05, this model includes additional **VEXP** (p-value=**0.0252**) in the set of significant explanatory variables at level 0.05, the coefficients of DATA, VEXP TURN and STOR have been increased, while the coefficient of LEXP and ln_size have been decreased, but the overall impact of observation No.17 on the original model is negative, therefore No.17 is an outlier.

After deleting observation No.55, the model becomes Equation (2-29):

Ln_effort = -18.377+ **1.203** *ln_size + 1.484*RELY + **1.763***DATA
+ **1.447***CPLX + 1.576*TIME + 0.147*STOR + 1.236*VIRT
-0.348*TURN+**2.530***ACAP+ 0.703*AEXP + 1.134*PCAP
+ **2.585***VEXP + 0.646*LEXP + **0.240***MODP + 1.389*TOOL
+ 1.894*SCED (2-29)

This model has RMSE equals 0.407, R-Square = 0.9632; Compared to Equation (2-27), where R-Square = 0.9576, and seven coefficients: ln_size, RELY, CPLX, TIME, ACAP, PCAP, SCED are significant to ln_effort at level of 0.05, this model includes additional **VEXP** (p-value=**0.0491**) in the set of significant explanatory variables at level 0.05, the coefficients of ln_size, DATA, VEXP, TURN and STOR have been significantly increased, while the coefficient of LEXP and has been decreased, but the overall impact of observation No.55 on the original model is negative, therefore, No.55 is an outlier.

After deleting observation No.56, the model becomes Equation (2-30):

Ln_effort = -16.378+ **1.158** *ln_size + 1.511*RELY + **1.739***DATA
+ 0.918*CPLX + 1.444*TIME + 0.327*STOR + **1.548***VIRT

-0.234*TURN+1.718*ACAP+ 0.846*AEXP + 1.361*PCAP

+ 1.923*VEXP + **0.0867***LEXP + 0.550*MODP + 1.490*TOOL

+ 2.0028*SCED (2-30)

This model has RMSE equals 0.41739, R-Square = 0.9613; Compared to Equation (2-27), where R-Square equals 0.9576, and seven coefficients: ln_size, RELY, CPLX, TIME, ACAP, PCAP, SCED are significant to ln_effort at level of 0.05, this model includes the same elements in the set of significant explanatory variables at level 0.05, the coefficients of DATA, TURN and VIRT have been significantly increased, while the coefficient of ln_size, LEXP have been decreased, but observation No.56 has little impact on the original model therefore, No.56 is NOT an outlier.

After deleting observations No.17, 55, and 56, the model becomes Equation (2-31):

Ln_effort = -17.093+ 1.164 *ln_size + 1.662*RELY + **2.259***DATA

+ **1.253***CPLX + 1.253*TIME + **0.609***STOR + **1.453***VIRT

-0.272*TURN+**2.453***ACAP+ 0.554*AEXP + **0.913***PCAP

+**2.797***VEXP–**0.712***LEXP + 0.496*MODP + 1.214*TOOL

+ 1.967*SCED (2-31)

This model has RMSE equals 0.34761, R-Square equals 0.9729; Compared with Equation (2-27)—the model with potential outliers present, where R-Square = 0.9576, and seven coefficients: ln_size, RELY, CPLX, TIME, ACAP, PCAP, SCED are significant to ln_effort at the level of 0.05, this model includes the additional elements DATA, VIRT, VEXP in the set of significant explanatory variables at level 0.05, the coefficients of DATA, CPLX, STOR, VIRT, TURN, ACAP, PCAP, LEXP and VEXP

have been significantly increased, the overall impact of observations No.17, 55 and 56 is very significant to the original model.

2.3.6 Variable Selection Results Using Forward Selection

Table 2-7 shows the summaries of the Forward Selection Procedure.

Table 2-7. Results of Forward Selection Procedure for Dependent Variable ln(Effort)

Step	Variable Entered	R^2 Value	Variables in Model	Estimated Coefficient	P-value
step 1	ln_size	0.7168	ln_size	1.07654	<0.0001
step 2	VIRT	0.8209	ln_size	1.15253	<0.0001
			VIRT	4.96783	<0.0001
Step 3	TIME	0.8599	TIME	2.49756	0.0002
			VIRT	3.43123	0.0001
			Ln_size	1.13151	<0.0001
Step 4	PCAP	0.8977	PCAP	2.22854	<0.0001
			VIRT	3.53071	<0.0001
			TIME	3.05498	<0.0001
			ln_size	1.10018	<0.0001
Step 5	SCED	0.9138	SCED	3.19650	0.0019
			PCAP	2.27153	<0.0001
			VIRT	3.08922	<0.0001
			TIME	3.03505	<0.0001
			ln_size	1.12092	<0.0001
Step 6	CPLX	0.9271	CPLX	1.36629	0.0023
			SCED	3.16133	0.0010
			PCAP	2.78768	<0.0001
			VIRT	2.98766	<0.0001
			TIME	2.38093	<0.0001
			ln_size	1.16837	<0.0001
Step 7	ACAP	0.9347	ACAP	1.63184	0.0143
			CPLX	1.69911	0.0002
			SCED	2.68653	0.0038
			PCAP	1.93169	0.0008
			VIRT	3.08625	<0.0001
			TIME	2.44299	<0.0001
			ln_size	1.21800	<0.0001
step 8	VEXP	0.9420	VEXP	2.42132	0.0119
			ACAP	1.98049	0.0026
			CPLX	1.65426	0.0002
			SCED	2.60165	0.0033
			PCAP	1.84038	0.0008
			VIRT	1.79934	0.0217
			TIME	2.56868	<0.0001
			ln_size	1.22346	<0.0001
step 9	RELY	0.9500	RELY	1.36987	0.0052
			VEXP	3.09210	0.0012
			PCAP	1.78497	0.0005
			ACAP	2.33603	0.0003
			SCED	2.36759	0.0044
			CPLX	1.27407	0.0027
			VIRT	1.51930	0.0395
			TIME	1.73009	0.0017
			ln_size	1.20180	<0.0001
step 10	MODP	0.9528	SCED	2.43220	0.0030
			MODP	0.93486	0.0851
			VEXP	3.06802	0.0011
			PCAP	1.41512	0.0089
			ACAP	2.22682	0.0004
			VIRT	1.27322	0.0828
			TIME	1.71011	0.0016
			CPLX	1.12838	0.0075
			RELY	1.51171	0.0022
			ln_size	1.18975	<0.0001

*No other variable met the 0.05000 significance level for entry into the model.

From Table 2-7 it can be seen that step 9 is the last step for the p-values for all the coefficients of all the 9 explanatory variables during the Forward Selection Procedure at the level significance of 0.05 (PIN). Step 10 to 14 can be continued if choosing PIN = 0.5. Other further steps for the procedure are listed in Table 2-8.

Table 2-8. Summary of Forward Selection

Step	Variable Entered	Vars In	Number R-Square	Partial R-Square	Model C(p)	F Value	Pr > F
1	ln_size	1	0.7168	0.7168	248.066	154.37	<.0001
2	VIRT	2	0.1041	0.8209	137.157	34.89	<.0001
3	TIME	3	0.0389	0.8599	96.9345	16.40	0.0002
4	PCAP	4	0.0379	0.8977	57.8774	21.48	<.0001
5	SCED	5	0.0161	0.9138	42.4140	10.66	0.0019
6	CPLX	6	0.0133	0.9271	30.0034	10.21	0.0023
7	ACAP	7	0.0076	0.9347	23.7715	6.40	0.0143
8	VEXP	8	0.0073	0.9420	17.8750	6.78	0.0119
9	RELY	9	0.0080	0.9500	11.1974	8.49	0.0052
10	MODP	10	0.0028	0.9528	10.1658	3.08	0.0851
11	DATA	11	0.0020	0.9548	10.0504	2.20	0.1442
12	TOOL	12	0.0014	0.9562	10.5348	1.59	0.2126
13	AEXP	13	0.0011	0.9573	11.3286	1.28	0.2642

The final acceptable reduced model is shown in Table 2-9.

Table 2-9. The Optimal Reduced Regression Model for Forward Selection Procedure

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	9	196.25094	21.80566	111.91	<.0001
Error	53	10.32712	0.19485		
Corrected Total	62	206.57806			

R-Square = 0.9500 and C(p) = 11.1974

Variable	Parameter Estimate	Standard Error	Type II SS	F Value	Pr > F
Intercept	-14.69863	1.21859	28.34937	145.49	<.0001
ln_size	1.20180	0.04650	130.13265	667.86	<.0001
RELY	1.36987	0.47025	1.65350	8.49	0.0052
CPLX	1.27407	0.40438	1.93427	9.93	0.0027
TIME	1.73009	0.52402	2.12400	10.90	0.0017
VIRT	1.51930	0.71973	0.86826	4.46	0.0395
ACAP	2.33603	0.60119	2.94193	15.10	0.0003
PCAP	1.78497	0.48495	2.63985	13.55	0.0005
VEXP	3.09210	0.90125	2.29362	11.77	0.0012
SCED	2.36759	0.79611	1.72335	8.84	0.0044

2.3.6 Variable Selection Results Using Backward Elimination

Table 2-10 and Table 2-11 show the summaries of the Backward Elimination Procedure.

Table 2-10. Results of Backward Elimination Procedure for Dependent Variable ln(Effort)

Step	Variable Removed	R^2 Value	Variables in Model	Estimated Coefficient	P-value
step 0		0.9576	ln_size	1.16263	<.0001
			RELY	1.53326	0.0054
			DATA	1.43248	0.1591
			CPLX	0.96500	0.0329
			TIME	1.62272	0.0057
			STOR	0.25883	0.6465
			VIRT	1.06389	0.2100
			TURN	-0.01419	0.9885
			ACAP	1.77751	0.0126
			AEXP	0.75896	0.2879
			PCAP	1.49816	0.0090
			VEXP	2.27808	0.1021
			LEXP	0.54306	0.8164
			MODP	0.51959	0.4634
			TOOL	1.56479	0.1434
			SCED	2.06952	0.0373
Step 1	TURN	0.9576	ln_size	1.16233	<.0001
			RELY	1.53321	0.0049
			DATA	1.43220	0.1546
			CPLX	0.96476	0.0310
			TIME	1.62361	0.0050
			STOR	0.25801	0.6423
			VIRT	1.06052	0.1890
			ACAP	1.77805	0.0115
			AEXP	0.75627	0.2675
			PCAP	1.49973	0.0071
			VEXP	2.28316	0.0870
			LEXP	0.54584	0.8129
			MODP	0.51533	0.4188
			TOOL	1.56112	0.1287
			SCED	2.06869	0.0351
Step 2	LEXP	0.9575	ln_size	1.16283	<.0001
			RELY	1.53225	0.0045
			DATA	1.39224	0.1560
			CPLX	0.97511	0.0270
			TIME	1.62236	0.0046
			STOR	0.28513	0.5963
			VIRT	1.07735	0.1761
			ACAP	1.75554	0.0109
			AEXP	0.73633	0.2716
			PCAP	1.49235	0.0067
			VEXP	2.47811	0.0175
			MODP	0.54571	0.3775
			TOOL	1.54188	0.1283
			SCED	2.15280	0.0177
Step 3	STOR	0.9573	ln_size	1.16162	<.0001
			RELY	1.62678	0.0014
			DATA	1.44783	0.1353
			CPLX	1.03083	0.0156
			TIME	1.72741	0.0013
			VIRT	1.19050	0.1188
			ACAP	1.81418	0.0074
			AEXP	0.74223	0.2642
			PCAP	1.44998	0.0072
			VEXP	2.36363	0.0193
			MODP	0.62232	0.2976
			TOOL	1.56314	0.1201
			SCED	1.99202	0.0184

Step 4	MODP	0.9563	ln_size	1.16747	<.0001
			RELY	1.59302	0.0017
			DATA	1.54570	0.1100
			CPLX	1.06682	0.0122
			TIME	1.73054	0.0012
			VIRT	1.20604	0.1144
			ACAP	1.89422	0.0050
			AEXP	0.69277	0.2961
			PCAP	1.64830	0.0012
			VEXP	2.37960	0.0186
			TOOL	1.98843	0.0323
			SCED	1.96164	0.0202
Step 5	AEXP	0.9553	ln_size	1.16086	<.0001
			RELY	1.51301	0.0024
			DATA	1.66929	0.0829
			CPLX	1.10849	0.0091
			TIME	1.72129	0.0013
			VIRT	1.16092	0.1279
			ACAP	2.18520	0.0005
			PCAP	1.56599	0.0018
			VEXP	2.84220	0.0021
			TOOL	1.59053	0.0583
			SCED	2.25510	0.0050
Step 6	VIRT	0.9532	ln_size	1.15781	<.0001
			RELY	1.68007	0.0007
			DATA	1.53923	0.1121
			CPLX	1.00810	0.0171
			TIME	1.85894	0.0005
			ACAP	2.28405	0.0003
			PCAP	1.52188	0.0026
			VEXP	3.64636	<.0001
			TOOL	2.04191	0.0114
			SCED	2.38731	0.0033
Step 7	**DATA**	**0.9509**	**ln_size**	**1.19789**	**<.0001**
			RELY	**1.73782**	**0.0006**
			CPLX	**0.97757**	**0.0222**
			TIME	**1.82230**	**0.0008**
			ACAP	**2.39591**	**0.0002**
			PCAP	**1.62295**	**0.0015**
			VEXP	**3.73305**	**<.0001**
			TOOL	**1.81828**	**0.0231**
			SCED	**2.47115**	**0.0027**

*All variables left in the model are significant at the 0.05000 level.

Table 2-11. Summary of Backward Elimination Procedure

Step	Variable Removed	Number Vars In	Partial R-Square	Model R-Square	C(p)	F Value	Pr > F
1	TURN	15	0.0000	0.9576	15.0002	0.00	0.9885
2	LEXP	14	0.0001	0.9575	13.0557	0.06	0.8129
3	STOR	13	0.0003	0.9573	11.3286	0.28	0.5963
4	MODP	12	0.0010	0.9563	10.3763	1.11	0.2976
5	AEXP	11	0.0010	0.9553	9.4325	1.11	0.2961
6	VIRT	10	0.0021	0.9532	9.7069	2.39	0.1279
7	DATA	9	0.0023	0.9509	10.2535	2.61	0.1121

From Table 2-10 and Table 2-11, it is clear that the *backward elimination selection procedure* is more efficient than the *forward selection procedure*, since it only takes 7 steps to find a reduced model with higher R-square value (0.9509) in *backward*

elimination selection procedure while in it takes 9 steps to find the reduced model with less R-square value (0.9500) in *forward selection procedure*, but missed the reduced model with higher R-square = 0.9509.

Table 2-12. Optimal Reduced Regression Model for Backward Elimination Procedure

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	9	196.43079	21.82564	114.00	<.0001
Error	53	10.14727	0.19146		
Corrected Total	62	206.57806			

R-Square = 0.9509 and C(p) = 10.2535

Variable	Parameter Estimate	Standard Error	Type II SS	F Value	Pr > F
Intercept	-15.81859	1.24040	31.13756	162.63	<.0001
ln_size	1.19789	0.04591	130.32332	680.69	<.0001
RELY	1.73782	0.47279	2.58673	13.51	0.0006
CPLX	0.97757	0.41490	1.06289	5.55	0.0222
TIME	1.82230	0.51142	2.43086	12.70	0.0008
ACAP	2.39591	0.59311	3.12418	16.32	0.0002
PCAP	1.62295	0.48435	2.14962	11.23	0.0015
VEXP	3.73305	0.72238	5.11292	26.71	<.0001
TOOL	1.81828	0.77713	1.04811	5.47	0.0231
SCED	2.47115	0.78480	1.89824	9.91	0.0027

2.3.7 Variable Selection Results Using Stepwise Selection (STEPWISE)

Table 2-13 shows the summaries of the Stepwise Procedure.

Table 2-13. Results of Stepwise Selection Procedure for Dependent Variable ln(Effort)

Step No.	Variable Entered Or removed	R^2 Value	Variables in Model	Estimated Coefficient	P-value
step 1	ln_size	0.7168	ln_size	1.07654	<0.0001
step 2	VIRT	0.8209	ln_size	1.15253	<0.0001
			VIRT	4.96783	<0.0001
Step 3	TIME	0.8599	ln_size	1.13151	<0.0001
			TIME	2.49756	0.0002
			VIRT	3.43123	0.0001
Step 4	PCAP	0.8977	ln_size	1.10018	<.0001
			TIME	3.05498	<.0001
			VIRT	3.53071	<.0001
			PCAP	2.22854	<.0001
Step 5	SCED	0.9138	ln_size	1.12092	<.0001
			TIME	3.03505	<.0001
			VIRT	3.08922	<.0001
			PCAP	2.27153	<.0001
			SCED	3.19650	0.0019
Step 6	CPLX	0.9271	ln_size	1.16837	<.0001
			CPLX	1.36629	0.0023
			TIME	2.38093	<.0001
			VIRT	2.98766	<.0001
			PCAP	2.78768	<.0001
			SCED	3.16133	0.0010
Step 7	ACAP	0.9347	ln_size	1.21800	<.0001
			CPLX	1.69911	0.0002
			TIME	2.44299	<.0001
			VIRT	3.08625	<.0001
			ACAP	1.63184	0.0143
			PCAP	1.93169	0.0008
			SCED	2.68653	0.0038
Step 8	VEXP	0.9420	ln_size	1.22346	<.0001
			CPLX	1.65426	0.0002
			TIME	2.56868	<.0001
			VIRT	1.79934	0.0217
			ACAP	1.98049	0.0026
			PCAP	1.84038	0.0008
			VEXP	2.42132	0.0119
			SCED	2.60165	0.0033
Step 9	RELY	0.9500	ln_size	1.20180	<.0001
			RELY	1.36987	0.0052
			CPLX	1.27407	0.0027
			TIME	1.73009	0.0017
			VIRT	1.51930	0.0395
			ACAP	2.33603	0.0003
			PCAP	1.78497	0.0005
			VEXP	3.09210	0.0012
			SCED	2.36759	0.0044

Step 10	MODP	0.9528	ln_size	1.18975	<.0001
			RELY	1.51171	0.0022
			CPLX	1.12838	0.0075
			TIME	1.71011	0.0016
			VIRT	1.27322	0.0828
			ACAP	2.22682	0.0004
			PCAP	1.41512	0.0089
			VEXP	3.06802	0.0011
			MODP	0.93486	0.0851
			SCED	2.43220	0.0030
Step 11	DATA	0.9548	ln_size	1.15390	<.0001
			RELY	1.41757	0.0038
			DATA	1.39599	0.1442
			CPLX	1.19083	0.0047
			TIME	1.73564	0.0013
			VIRT	1.43118	0.0521
			ACAP	2.12171	0.0007
			PCAP	1.34665	0.0119
			VEXP	2.93063	0.0016
			MODP	0.92139	0.0862
			SCED	2.34560	0.0039

*All variables left in the model are significant at the 0.1500 level.

No other variable met the 0.1500 significance level for entry into the model.

Table 2-14. Summary of Stepwise Selection

Step	Variable Entered	Variable Removed	Vars In	Number R-Square	Partial R-Square	Model C(p)	F Value	Pr > F
1	ln_size		1	0.7168	0.7168	248.066	154.37	<.0001
2	VIRT		2	0.1041	0.8209	137.157	34.89	<.0001
3	TIME		3	0.0389	0.8599	96.9345	16.40	0.0002
4	PCAP		4	0.0379	0.8977	57.8774	21.48	<.0001
5	SCED		5	0.0161	0.9138	42.4140	10.66	0.0019
6	CPLX		6	0.0133	0.9271	30.0034	10.21	0.0023
7	ACAP		7	0.0076	0.9347	23.7715	6.40	0.0143
8	VEXP		8	0.0073	0.9420	17.8750	6.78	0.0119
9	RELY		9	0.0080	0.9500	11.1974	8.49	0.0052
10	MODP		10	0.0028	0.9528	10.1658	3.08	0.0851
11	DATA		11	0.0020	0.9548	10.0504	2.20	0.1442

From Table 2-13 and Table 2-14, it is clear that the *backward elimination selection procedure* is more efficient than the *stepwise selection procedure*, which is very similar to *forward selection procedure* since the *backward elimination selection procedure* only takes 7 steps to find a reduced model with higher R-square value (0.9509) while in *stepwise selection procedure* it takes 9 steps to find the reduced model with less R-square value (0.9500), again *stepwise selection procedure* missed the reduced model with higher R-square = 0.9509.

2.3.8 Variable Selection Results Using Maximum R^2 Improvement (MAXR)

The maximum R^2 improvement technique does not settle on a single model. Instead, it tries to find the "best" one-variable model, the "best" two-variable model, and so forth, although it is not guaranteed to find the model with the largest R^2 for each size. The MAXR method begins by finding the one-variable model producing the highest R^2. Then another variable, the one that yields the greatest increase in R^2, is added. Once the two-variable model is obtained, each of the variables in the model is compared with each variable not in the model. For each comparison, the MAXR method determines if removing one variable and replacing it with the other variable increases R^2. After comparing all possible switches, the MAXR method makes the switch that produces the largest increase in R^2. Comparisons begin again, and the process continues until the MAXR method finds that no switch can further increase R^2. Thus, the two-variable model achieved is considered the "best" two-variable model the technique can find. Another variable is then added to the model, and the comparing-and-switching process is repeated to find the "best" three-variable model, and so forth.

The difference between the STEPWISE method and the MAXR method is that all switches are evaluated before any switch is made in the MAXR method. In the STEPWISE method, the "worst" variable may be removed without considering what adding the "best" remaining variable might accomplish. The MAXR method may require much more computer time than the STEPWISE method.

Table 2-15 shows the summaries of the Maximum R^2 Improvement Procedure.

Table 2-15. Results of Maximum R^2 Improvement Procedure for Dependent Variable ln(Effort)

Level	Variables in Model	R^2	C(p)
1	ln_size	0.7168	248.0657
2	ln_size, 6	0.8209	137.1573
3	ln_size, 6, 4	0.8599	96.9345
4	ln_size, 6, 4, 10	0.8977	57.8774
5	ln_size, 6, 4, 10, 15	0.9138	42.4140
6	ln_size, 6, 4, 10, 15, 3	0.9271	30.0034
7	ln_size, 6, 4, 10, 15, 3, 8	0.9347	23.7715
8	ln_size, 4, 10, 15, 3, 8, 11	0.9360	22.3824
9	ln_size, 4, 10, 15, 1, 8, 11	0.9370	21.2729
10	ln_size, 4, 13, 15, 1, 8, 11	0.9393	18.7534
11	ln_size, 4, 13, 15, 1, 8, 11, 3	0.9440	15.6914
12	ln_size, 4, 10, 15, 1, 8, 11, 3	0.9458	13.7541
13	**ln_size, 4, 10, 15, 1, 8, 11, 3, 14**	0.9509	10.2535
14	ln_size, 4, 10, 15, 1, 8, 11, 3, 14, 2	0.9532	9.7069
15	ln_size, 4, 10, 15, 1, 8, 11, 3, 14, 2, 6	0.9553	9.4325
16	ln_size, 4, 10, 15, 1, 8, 11, 3, 14, 2, 6, 9	0.9563	10.3763
17	ln_size, 4, 10, 15, 1, 8, 11, 3, 14, 2, 6, 9, 13	0.9573	11.3286
18	ln_size, 4, 10, 15, 1, 8, 11, 3, 14, 2, 6, 9, 13, 5	0.9575	13.0557
19	ln_size, 4, 10, 15, 1, 8, 11, 3, 14, 2, 6, 9, 13, 5, 12	0.9576	15.0002
20	ln_size, 4, 10, 15, 1, 8, 11, 3, 14, 2, 6, 9, 13, 5, 12, 7	0.9576	17.0000

Table 2-16. Final Reduced Regression Model of Maximum R^2 Improvement Procedure for Dependent Variable ln(Effort)

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	9	196.43079	21.82564	114.00	<.0001
Error	53	10.14727	0.19146		
Corrected Total	62	206.57806			

Variable TOOL Entered: R-Square = 0.9509 and C(p) = 10.2535

Variable	Parameter Estimate	Standard Error	Type II SS	F Value	Pr > F
Intercept	-15.81859	1.24040	31.13756	162.63	<.0001
ln_size	1.19789	0.04591	130.32332	680.69	<.0001
RELY	1.73782	0.47279	2.58673	13.51	0.0006
CPLX	0.97757	0.41490	1.06289	5.55	0.0222
TIME	1.82230	0.51142	2.43086	12.70	0.0008
ACAP	2.39591	0.59311	3.12418	16.32	0.0002
PCAP	1.62295	0.48435	2.14962	11.23	0.0015
VEXP	3.73305	0.72238	5.11292	26.71	<.0001
TOOL	1.81828	0.77713	1.04811	5.47	0.0231
SCED	2.47115	0.78480	1.89824	9.91	0.0027

Table 2-16 and Table 17 show that it only takes 13 steps to find a reduced model with R-square value of 0.9509, it is efficient compared with Minimum R^2 Improvement.

2.3.9 Variable Selection Results Using Minimum R^2 Improvement (MINR)

The MINR method closely resembles the MAXR method, but the switch chosen is the one that produces the smallest increase in R^2. For a given number of variables in the

model, the MAXR and MINR methods usually produce the same "best" model, but the MINR method considers more models of each size.

The starting point at each level is the maximum R^2 model from the previous level. The reason for the choice of the model with minimum R^2 at each step within any level is to show the user a large number of reasonable models at each level. Typically for each switch the results of the regression are provided. Both R^2 improvement procedures are designed to produce a large number of possible regressions at each level. These two methods will tend to yield more possible solutions than the stepwise methods. In addition the minimum R^2 method usually produces more solutions than the maximum R^2 method. In our experiment, the maximum R^2 method produces 20 steps while the minimum R^2 method produces 156 steps, not until step 123 can we find the optimal reduced regression model with 9 explanatory variables, R-Square is 0.9509 in the minimum R^2 method. Table 2-17 summarizes the results of the minimum R^2 method on our transformed COCOMO dataset.

Table 2-17. Summary of the Minimum R^2 Improvement Method

Number in Model	R-Square	Adjusted R-Square	C(p)	MSE	S(p)	Variables in Model
1	0.7168	0.7121	248.0657	0.95919	0.01599	ln_size
1	0.2640	0.2519	738.9514	2.49257	0.04154	2
1	0.1181	0.1036	897.1347	2.98669	0.04978	7
1	0.0988	0.0840	918.0604	3.05206	0.05087	4
1	0.0960	0.0812	921.0367	3.06135	0.05102	1
1	0.0522	0.0367	968.5396	3.20974	0.05350	5
1	0.0492	0.0336	971.8104	3.21996	0.05367	13
1	0.0430	0.0273	978.5432	3.24099	0.05402	10
1	0.0266	0.0107	996.2654	3.29635	0.05494	6
1	0.0232	0.0072	999.9489	3.30785	0.05513	11
1	0.0201	0.0040	1003.355	3.31849	0.05531	12
1	0.0081	-.0082	1016.352	3.35909	0.05598	9
1	0.0043	-.0120	1020.422	3.37181	0.05620	8
1	0.0033	-.0130	1021.549	3.37533	0.05626	3
1	0.0026	-.0137	1022.272	3.37759	0.05629	15
1	0.0005	-.0159	1024.632	3.38496	0.05642	14
2	0.8209	0.8149	137.1573	0.61660	0.01045	ln_size, 6
2	0.8203	0.8143	137.8249	0.61872	0.01049	ln_size, 4
2	0.7961	0.7894	164.0036	0.70186	0.01190	ln_size, 5
2	0.7961	0.7893	164.0580	0.70203	0.01190	ln_size, 12
2	0.7853	0.7782	175.7256	0.73908	0.01253	ln_size, 1
2	0.7824	0.7751	178.9140	0.74921	0.01270	ln_size, 11
2	0.7655	0.7577	197.2502	0.80744	0.01369	ln_size, 14
2	0.7594	0.7514	203.8680	0.82846	0.01404	ln_size, 3
2	0.7551	0.7469	208.5159	0.84322	0.01429	ln_size, 15
2	0.7458	0.7373	218.6397	0.87537	0.01484	ln_size, 13
2	0.7341	0.7253	231.2196	0.91532	0.01551	ln_size, 7
2	0.7265	0.7174	239.4968	0.94161	0.01596	ln_size, 10
2	0.7239	0.7147	242.3580	0.95069	0.01611	ln_size, 9
2	0.7206	0.7113	245.9156	0.96199	0.01630	ln_size, 2
2	0.7203	0.7110	246.2420	0.96303	0.01632	ln_size, 8
2	0.3980	0.3779	595.6841	2.07278	0.03513	2, 4
3	0.8599	0.8527	96.9345	0.49069	0.00846	ln_size, 4, 6
3	0.8590	0.8518	97.9090	0.49384	0.00851	ln_size, 4, 13
3	0.8559	0.8485	101.2510	0.50463	0.00870	ln_size, 4, 10
3	0.8508	0.8432	106.7642	0.52243	0.00901	ln_size, 4, 8
3	0.8508	0.8432	106.7888	0.52251	0.00901	ln_size, 4, 14
3	0.8490	0.8414	108.6559	0.52854	0.00911	ln_size, 1, 6
3	0.8484	0.8407	109.3253	0.53071	0.00915	ln_size, 4, 11
3	0.8476	0.8398	110.2616	0.53373	0.00920	ln_size, 5, 6
3	0.8453	0.8374	112.7039	0.54162	0.00934	ln_size, 4, 15
3	0.8442	0.8362	113.9412	0.54561	0.00941	ln_size, 4, 12
3	0.8432	0.8352	115.0350	0.54915	0.00947	ln_size, 5, 11
3	0.8429	0.8350	115.2645	0.54989	0.00948	ln_size, 5, 15
3	0.8423	0.8343	115.9720	0.55217	0.00952	ln_size, 6, 10
3	0.8401	0.8319	118.4014	0.56002	0.00966	ln_size, 1, 13
3	0.8380	0.8298	120.6265	0.56720	0.00978	ln_size, 4, 9
3	0.8375	0.8293	121.1435	0.56887	0.00981	ln_size, 1, 14
4	0.9032	0.8966	51.9164	0.34468	0.00605	ln_size, 1, 8, 11
4	0.8991	0.8921	56.4422	0.35955	0.00631	ln_size, 4, 8, 12
4	0.8980	0.8909	57.6321	0.36346	0.00638	ln_size, 1, 8, 12
4	0.8977	0.8907	57.8774	0.36426	0.00639	ln_size, 4, 6, 10
4	0.8961	0.8890	59.5990	0.36992	0.00649	ln_size, 4, 8, 11
4	0.8949	0.8876	60.9795	0.37446	0.00657	ln_size, 4, 10, 12
4	0.8938	0.8864	62.1632	0.37834	0.00664	ln_size, 4, 6, 8
4	0.8928	0.8854	63.2318	0.38185	0.00670	ln_size, 1, 6, 8
4	0.8927	0.8853	63.3524	0.38225	0.00671	ln_size, 1, 6, 10
4	0.8926	0.8852	63.3826	0.38235	0.00671	ln_size, 4, 10, 11
4	0.8886	0.8810	67.7346	0.39665	0.00696	ln_size, 1, 10, 11
4	0.8853	0.8774	71.3333	0.40847	0.00717	ln_size, 1, 10, 12
4	0.8850	0.8771	71.6715	0.40958	0.00719	ln_size, 4, 6, 13
4	0.8840	0.8760	72.7759	0.41321	0.00725	ln_size, 4, 13, 15
4	0.8830	0.8750	73.7936	0.41655	0.00731	ln_size, 4, 10, 15
4	0.8828	0.8748	74.0090	0.41726	0.00732	ln_size, 3, 6, 10

Table 2-17. Summary of the Minimum R^2 Improvement Method (cont.)

Number in Model	R-Square	Adjusted R-Square	C(p)	MSE	S(p)	Variables in Model
5	0.9187	0.9116	37.1373	0.29464	0.00526	ln_size, 1, 4, 8, 11
5	0.9182	0.9110	37.6815	0.29646	0.00529	ln_size, 1, 8, 11, 14
5	0.9165	0.9092	39.5181	0.30260	0.00540	ln_size, 1, 8, 11, 13
5	0.9156	0.9082	40.5414	0.30602	0.00546	ln_size, 5, 10, 11, 15
5	0.9138	0.9063	42.4140	0.31228	0.00558	ln_size, 4, 6, 10, 15
5	0.9136	0.9060	42.6645	0.31311	0.00559	ln_size, 3, 4, 8, 11
5	0.9134	0.9058	42.8729	0.31381	0.00560	ln_size, 1, 3, 8, 11
5	0.9129	0.9053	43.3776	0.31550	0.00563	ln_size, 1, 4, 8, 12
5	0.9128	0.9052	43.5207	0.31598	0.00564	ln_size, 4, 10, 11, 15
5	0.9121	0.9044	44.2778	0.31851	0.00569	ln_size, 3, 4, 6, 8
5	0.9117	0.9040	44.6976	0.31991	0.00571	ln_size, 1, 6, 8, 11
5	0.9116	0.9038	44.8610	0.32046	0.00572	ln_size, 1, 8, 12, 14
5	0.9114	0.9037	45.0090	0.32095	0.00573	ln_size, 1, 5, 8, 11
5	0.9114	0.9036	45.0822	0.32120	0.00574	ln_size, 3, 4, 6, 10
5	0.9113	0.9035	45.1745	0.32150	0.00574	ln_size, 1, 7, 8, 11
5	0.9106	0.9028	45.8730	0.32384	0.00578	ln_size, 1, 8, 11, 15
6	0.9303	0.9228	26.6051	0.25726	0.00468	ln_size, 4, 5, 10, 11, 15
6	0.9302	0.9227	26.6510	0.25741	0.00468	ln_size, 1, 4, 8, 11, 13
6	0.9300	0.9225	26.8559	0.25811	0.00469	ln_size, 1, 4, 8, 11, 14
6	0.9276	0.9198	29.5183	0.26717	0.00486	ln_size, 1, 5, 10, 11, 15
6	0.9271	0.9193	30.0034	0.26882	0.00489	ln_size, 3, 4, 6, 10, 15
6	0.9270	0.9192	30.1609	0.26935	0.00490	ln_size, 1, 4, 9, 10, 14
6	0.9265	0.9186	30.6778	0.27111	0.00493	ln_size, 1, 4, 8, 10, 11
6	0.9265	0.9186	30.7303	0.27129	0.00493	ln_size, 1, 3, 4, 8, 11
6	0.9263	0.9184	30.9488	0.27203	0.00495	ln_size, 1, 4, 8, 11, 15
6	0.9259	0.9179	31.3561	0.27342	0.00497	ln_size, 1, 4, 7, 8, 11
6	0.9256	0.9176	31.6473	0.27441	0.00499	ln_size, 1, 8, 11, 13, 15
6	0.9248	0.9168	32.4799	0.27724	0.00504	ln_size, 1, 4, 10, 11, 15
6	0.9248	0.9168	32.5057	0.27733	0.00504	ln_size, 1, 8, 11, 14, 15
6	0.9243	0.9162	33.0808	0.27929	0.00508	ln_size, 1, 5, 8, 11, 15
6	0.9239	0.9157	33.5540	0.28090	0.00511	ln_size, 3, 4, 6, 8, 10
6	0.9237	0.9156	33.6665	0.28128	0.00511	ln_size, 1, 8, 10, 11, 14
7	0.9393	0.9316	18.7534	0.22780	0.00422	ln_size, 1, 4, 8, 11, 13, 15
7	0.9370	0.9290	21.2729	0.23653	0.00438	ln_size, 1, 4, 8, 10, 11, 15
7	0.9369	0.9288	21.4526	0.23715	0.00439	ln_size, 1, 4, 8, 11, 14, 15
7	0.9363	0.9282	22.0655	0.23928	0.00443	ln_size, 1, 4, 8, 10, 11, 14
7	0.9360	0.9279	22.3824	0.24037	0.00445	ln_size, 3, 4, 8, 10, 11, 15
7	0.9359	0.9278	22.4673	0.24067	0.00446	ln_size, 1, 5, 8, 10, 11, 15
7	0.9359	0.9277	22.4895	0.24075	0.00446	ln_size, 1, 3, 4, 8, 10, 11
7	0.9357	0.9275	22.7449	0.24163	0.00447	ln_size, 1, 3, 4, 8, 11, 13
7	0.9355	0.9273	22.9516	0.24235	0.00449	ln_size, 1, 4, 6, 9, 10, 14
7	0.9352	0.9270	23.2181	0.24327	0.00450	ln_size, 1, 4, 9, 10, 12, 14
7	0.9348	0.9265	23.7289	0.24504	0.00454	ln_size, 4, 5, 8, 10, 11, 15
7	0.9347	0.9264	23.7715	0.24519	0.00454	ln_size, 3, 4, 6, 8, 10, 15
7	0.9344	0.9261	24.0677	0.24621	0.00456	ln_size, 1, 4, 5, 10, 11, 15
7	0.9344	0.9261	24.0765	0.24624	0.00456	ln_size, 3, 4, 5, 10, 11, 15
7	0.9344	0.9260	24.1562	0.24652	0.00457	ln_size, 1, 4, 7, 8, 10, 11
7	0.9340	0.9256	24.5253	0.24780	0.00459	ln_size, 1, 3, 4, 8, 11, 14
8	0.9458	0.9378	13.7541	0.20732	0.00391	ln_size, 1, 3, 4, 8, 10, 11, 15
8	0.9457	0.9377	13.8316	0.20760	0.00392	ln_size, 1, 4, 8, 10, 11, 14, 15
8	0.9440	0.9357	15.6914	0.21416	0.00404	ln_size, 1, 3, 4, 8, 11, 13, 15
8	0.9436	0.9353	16.1058	0.21562	0.00407	ln_size, 1, 4, 8, 10, 11, 13, 15
8	0.9433	0.9349	16.5163	0.21707	0.00410	ln_size, 3, 4, 5, 8, 10, 11, 15
8	0.9432	0.9348	16.6074	0.21739	0.00410	ln_size, 1, 4, 5, 8, 10, 11, 15
8	0.9430	0.9346	16.7435	0.21787	0.00411	ln_size, 1, 4, 7, 8, 10, 11, 15
8	0.9420	0.9334	17.8412	0.22174	0.00418	ln_size, 1, 4, 8, 11, 13, 14, 15
8	0.9420	0.9334	17.8750	0.22186	0.00419	ln_size, 3, 4, 6, 8, 10, 11, 15
8	0.9417	0.9331	18.2138	0.22306	0.00421	ln_size, 1, 4, 8, 9, 10, 12, 14
8	0.9417	0.9331	18.2156	0.22307	0.00421	ln_size, 1, 3, 4, 8, 10, 11, 14
8	0.9417	0.9330	18.2416	0.22316	0.00421	ln_size, 1, 3, 4, 6, 8, 10, 11
8	0.9415	0.9328	18.4125	0.22376	0.00422	ln_size, 1, 4, 8, 9, 10, 11, 14
8	0.9414	0.9328	18.4900	0.22403	0.00423	ln_size, 1, 3, 5, 8, 10, 11, 15
8	0.9414	0.9327	18.5583	0.22427	0.00423	ln_size, 1, 5, 8, 10, 11, 14, 15
8	0.9413	0.9326	18.6539	0.22461	0.00424	ln_size, 1, 3, 4, 7, 8, 10, 11

Table 2-17. Summary of the Minimum R^2 Improvement Method (cont.)

Number In Model	R^2	Adjusted R^2	C(p)	MSE	S(p)	Variables in Model
9	**0.9509**	**0.9425**	**10.2535**	**0.19146**	**0.00368**	**ln_size, 1, 3, 4, 8, 10, 11, 14, 15**
9	**0.9500**	**0.9415**	**11.1974**	**0.19485**	**0.00375**	**ln_size, 1, 3, 4, 6, 8, 10, 11, 15**
9	**0.9500**	**0.9415**	**11.2446**	**0.19502**	**0.00375**	**ln_size, 1, 3, 4, 8, 10, 11, 13, 15**
9	0.9498	0.9412	11.4626	0.19580	0.00377	ln_size, 1, 3, 4, 7, 8, 10, 11, 15
9	0.9489	0.9402	12.4478	0.19935	0.00383	ln_size, 1, 3, 4, 5, 8, 10, 11, 15
9	0.9485	0.9398	12.8003	0.20061	0.00386	ln_size, 1, 4, 5, 8, 10, 11, 14, 15
9	0.9478	0.9389	13.6263	0.20358	0.00392	ln_size, 1, 2, 4, 8, 10, 11, 14, 15
9	0.9475	0.9385	13.9627	0.20479	0.00394	ln_size, 1, 4, 8, 9, 10, 11, 14, 15
9	0.9473	0.9383	14.1658	0.20552	0.00395	ln_size, 1, 4, 7, 8, 10, 11, 14, 15
9	0.9472	0.9383	14.1931	0.20562	0.00395	ln_size, 1, 4, 8, 10, 11, 13, 14, 15
9	0.9470	0.9380	14.4242	0.20645	0.00397	ln_size, 1, 2, 3, 4, 8, 10, 11, 15
9	0.9466	0.9376	14.8722	0.20806	0.00400	ln_size, 1, 4, 6, 8, 10, 11, 14, 15
9	0.9465	0.9374	15.0085	0.20855	0.00401	ln_size, 1, 3, 4, 8, 10, 11, 12, 15
9	0.9464	0.9373	15.0696	0.20877	0.00401	ln_size, 1, 4, 5, 8, 10, 11, 13, 15
9	0.9462	0.9371	15.2756	0.20951	0.00403	ln_size, 1, 4, 8, 10, 11, 12, 14, 15
9	0.9461	0.9369	15.4446	0.21012	0.00404	ln_size, 1, 4, 5, 7, 8, 10, 11, 15
10	0.9532	0.9442	9.7069	0.18581	0.00364	ln_size, 1, 2, 3, 4, 8, 10, 11, 14, 15
10	0.9528	0.9437	10.1658	0.18749	0.00368	ln_size, 1, 3, 4, 6, 8, 10, 11, 13, 15
10	0.9526	0.9435	10.4039	0.18836	0.00369	ln_size, 1, 3, 4, 6, 8, 10, 11, 14, 15
10	0.9524	0.9432	10.6155	0.18914	0.00371	ln_size, 1, 3, 4, 5, 8, 10, 11, 14, 15
10	0.9521	0.9429	10.8938	0.19016	0.00373	ln_size, 1, 3, 4, 7, 8, 10, 11, 14, 15
10	0.9521	0.9429	10.9598	0.19040	0.00373	ln_size, 1, 3, 4, 8, 9, 10, 11, 14, 15
10	0.9520	0.9428	10.9943	0.19053	0.00374	ln_size, 1, 2, 3, 4, 6, 8, 10, 11, 15
10	0.9520	0.9428	11.0017	0.19055	0.00374	ln_size, 1, 3, 4, 8, 10, 11, 13, 14, 15
10	0.9517	0.9425	11.3139	0.19170	0.00376	ln_size, 1, 3, 4, 6, 7, 8, 10, 11, 15
10	0.9513	0.9420	11.7724	0.19338	0.00379	ln_size, 1, 3, 4, 5, 6, 8, 10, 11, 15
10	0.9513	0.9419	11.8091	0.19351	0.00379	ln_size, 1, 3, 4, 5, 8, 10, 11, 13, 15
10	0.9512	0.9419	11.8552	0.19368	0.00380	ln_size, 1, 2, 3, 4, 8, 10, 11, 13, 15
10	0.9512	0.9418	11.9481	0.19402	0.00380	ln_size, 1, 3, 4, 5, 7, 8, 10, 11, 15
10	0.9511	0.9417	11.9973	0.19420	0.00381	ln_size, 1, 3, 4, 7, 8, 10, 11, 13, 15
10	0.9510	0.9416	12.0704	0.19447	0.00381	ln_size, 1, 3, 4, 8, 10, 11, 12, 14, 15
10	0.9510	0.9416	12.1002	0.19458	0.00382	ln_size, 1, 2, 3, 4, 7, 8, 10, 11, 15
11	0.9553	0.9457	9.4325	0.18095	0.00362	ln_size, 1, 2, 3, 4, 6, 8, 10, 11, 14, 15
11	0.9548	0.9450	10.0504	0.18326	0.00367	ln_size, 1, 2, 3, 4, 6, 8, 10, 11, 13, 15
11	0.9544	0.9446	10.4432	0.18473	0.00369	ln_size, 1, 2, 3, 4, 5, 8, 10, 11, 14, 15
11	0.9542	0.9444	10.6214	0.18540	0.00371	ln_size, 1, 2, 3, 4, 7, 8, 10, 11, 14, 15
11	0.9541	0.9442	10.7406	0.18584	0.00372	ln_size, 1, 2, 3, 4, 8, 10, 11, 13, 14, 15
11	0.9540	0.9441	10.8230	0.18615	0.00372	ln_size, 1, 2, 3, 4, 8, 9, 10, 11, 14, 15
11	0.9540	0.9441	10.8853	0.18638	0.00373	ln_size, 1, 2, 3, 6, 8, 9, 10, 11, 14, 15
11	0.9537	0.9437	11.1993	0.18755	0.00375	ln_size, 1, 3, 4, 6, 8, 10, 11, 13, 14, 15
11	0.9536	0.9436	11.2609	0.18778	0.00376	ln_size, 1, 2, 3, 4, 8, 10, 11, 12, 14, 15
11	0.9536	0.9436	11.3144	0.18798	0.00376	ln_size, 1, 3, 4, 5, 8, 9, 10, 11, 14, 15
11	0.9536	0.9436	11.3152	0.18799	0.00376	ln_size, 1, 2, 3, 4, 6, 7, 8, 10, 11, 15
11	0.9535	0.9434	11.4511	0.18850	0.00377	ln_size, 1, 3, 4, 6, 8, 9, 10, 11, 13, 15
11	0.9534	0.9434	11.4763	0.18859	0.00377	ln_size, 1, 3, 4, 5, 6, 8, 10, 11, 14, 15
11	0.9534	0.9433	11.5437	0.18884	0.00378	ln_size, 1, 3, 4, 8, 9, 10, 11, 13, 14, 15
11	0.9533	0.9432	11.6230	0.18914	0.00378	ln_size, 1, 3, 4, 5, 6, 8, 10, 11, 13, 15
11	0.9532	0.9432	11.6841	0.18937	0.00379	ln_size, 1, 3, 4, 6, 7, 8, 10, 11, 14, 15
12	0.9563	0.9458	10.3763	0.18055	0.00368	ln_size, 1, 2, 3, 4, 6, 8, 9, 10, 11, 14, 15
12	0.9562	0.9456	10.5348	0.18115	0.00370	ln_size, 1, 2, 3, 4, 6, 8, 10, 11, 13, 14, 15
12	0.9558	0.9452	10.8740	0.18244	0.00372	ln_size, 1, 2, 3, 4, 5, 6, 8, 10, 11, 14, 15
12	0.9557	0.9451	10.9852	0.18287	0.00373	ln_size, 1, 2, 3, 4, 6, 7, 8, 10, 11, 14, 15
12	0.9555	0.9448	11.2375	0.18383	0.00375	ln_size, 1, 2, 3, 4, 6, 8, 10, 11, 12, 14, 15
12	0.9553	0.9445	11.5090	0.18486	0.00377	ln_size, 1, 3, 4, 6, 8, 9, 10, 11, 13, 14, 15
12	0.9552	0.9445	11.5254	0.18493	0.00377	ln_size, 1, 2, 3, 4, 5, 8, 9, 10, 11, 14, 15
12	0.9551	0.9443	11.6944	0.18557	0.00379	ln_size, 1, 2, 3, 4, 6, 8, 9, 10, 11, 13, 15
12	0.9551	0.9443	11.7079	0.18562	0.00379	ln_size, 1, 2, 3, 4, 5, 6, 8, 9, 10, 13, 15
12	0.9551	0.9443	11.7117	0.18564	0.00379	ln_size, 1, 2, 3, 4, 8, 9, 10, 11, 13, 14, 15
12	0.9550	0.9442	11.7957	0.18596	0.00380	ln_size, 1, 2, 3, 4, 5, 7, 8, 10, 11, 14, 15
12	0.9550	0.9442	11.7975	0.18596	0.00380	ln_size, 1, 2, 3, 4, 6, 7, 8, 10, 11, 13, 15
12	0.9549	0.9441	11.8914	0.18632	0.00380	ln_size, 1, 2, 3, 4, 5, 8, 10, 11, 13, 14, 15
12	0.9548	0.9440	11.9936	0.18671	0.00381	ln_size, 1, 3, 4, 5, 6, 8, 9, 10, 11, 14, 15
12	0.9548	0.9439	12.0220	0.18682	0.00381	ln_size, 1, 2, 3, 4, 6, 8, 10, 11, 12, 13, 15
12	0.9547	0.9439	12.0592	0.18696	0.00382	ln_size, 1, 2, 3, 4, 7, 8, 9, 10, 11, 13, 15

Table 2-17. Summary of the Minimum R^2 Improvement Method (cont.)

Number In Model	R^2	Adjusted R^2	C(p)	MSE	S(p)	Variables in Model
13	0.9573	0.9459	11.3286	0.18016	0.00375	ln_size, 1,2,3,4,6,8,9,10,11,13,14,15
13	0.9568	0.9454	11.8171	0.18206	0.00379	ln_size, 1,2,3,4,5,6,8,9,10,11,14,15
13	0.9566	0.9451	12.0367	0.18291	0.00381	ln_size, 1,2,3,4,6,8,9,10,11,12,14,15
13	0.9565	0.9449	12.1702	0.18343	0.00382	ln_size, 1,2,3,4,6,7,8,9,10,11,14,15
13	0.9564	0.9449	12.2424	0.18371	0.00383	ln_size, 1,2,3,4,5,6,8,10,11,13,14,15
13	0.9562	0.9446	12.4385	0.18447	0.00384	ln_size, 1,2,3,4,6,7,8,10,11,13,14,15
13	0.9562	0.9446	12.4923	0.18468	0.00385	ln_size, 1,2,3,4,6,8,10,11,12,13,14,15
13	0.9561	0.9445	12.5732	0.18500	0.00385	ln_size, 1,2,3,4,5,6,7,8,10,11,14,15
13	0.9559	0.9442	12.8032	0.18589	0.00387	ln_size, 1,2,3,4,6,7,8,10,11,12,14,15
13	0.9559	0.9442	12.8050	0.18590	0.00387	ln_size, 1,2,3,4,5,6,8,10,11,12,14,15
13	0.9558	0.9441	12.8646	0.18613	0.00388	ln_size, 1,2,3,4,5,8,9,10,11,13,14,15
13	0.9557	0.9439	13.0484	0.18685	0.00389	ln_size, 1,3,4,5,6,8,9,10,11,13,14,15
13	0.9556	0.9438	13.1402	0.18720	0.00390	ln_size, 1,2,3,4,5,7,8,9,10,11,14,15
13	0.9555	0.9437	13.2676	0.18770	0.00391	ln_size, 1,2,3,4,5,8,9,10,11,12,14,15
13	0.9554	0.9436	13.3539	0.18803	0.00392	ln_size, 1,2,3,4,5,6,8,9,10,11,13,15
13	0.9554	0.9435	13.3964	0.18820	0.00392	ln_size, 1,2,3,4,8,9,10,11,12,13,14,15
14	0.9575	0.9451	13.0557	0.18283	0.00389	ln_size, 1,2,3,4,5,6,8,9,10,11,13,14,15
14	0.9574	0.9449	13.2142	0.18346	0.00390	ln_size, 1,2,3,4,6,8,9,10,11,12,13,14,15
14	0.9573	0.9448	13.3285	0.18391	0.00391	ln_size, 1,2,3,4,6,7,8,9,10,11,13,14,15
14	0.9570	0.9444	13.6515	0.18519	0.00394	ln_size, 1,2,3,4,5,6,8,9,10,11,12,14,15
14	0.9569	0.9443	13.7100	0.18543	0.00395	ln_size, 1,2,3,4,5,6,7,8,9,10,11,14,15
14	0.9568	0.9442	13.8575	0.18601	0.00396	ln_size, 1,2,3,4,6,7,8,9,10,11,12,14,15
14	0.9565	0.9438	14.1724	0.18726	0.00398	ln_size, 1,2,3,4,5,6,7,8,10,11,13,14,15
14	0.9564	0.9437	14.2328	0.18750	0.00399	ln_size, 1,2,3,4,5,6,8,10,11,12,13,14,15
14	0.9563	0.9435	14.3830	0.18810	0.00400	ln_size, 1,2,3,4,6,7,8,10,11,12,13,14,15
14	0.9562	0.9434	14.4978	0.18855	0.00401	ln_size, 1,2,3,4,5,6,7,8,10,11,12,14,15
14	0.9560	0.9431	14.7393	0.18951	0.00403	ln_size, 1,2,3,4,5,8,9,10,11,12,13,14,15
14	0.9560	0.9431	14.7552	0.18957	0.00403	ln_size, 1,2,3,4,5,7,8,9,10,11,13,14,15
14	0.9558	0.9429	14.8991	0.19015	0.00405	ln_size, 1,2,3,4,5,7,8,9,10,11,12,14,15
14	0.9557	0.9428	15.0482	0.19074	0.00406	ln_size, 1,3,4,5,6,7,8,9,10,11,13,14,15
14	0.9557	0.9428	15.0484	0.19074	0.00406	ln_size, 1,3,4,5,6,8,9,10,11,12,13,14,15
14	0.9556	0.9426	15.1643	0.19120	0.00407	ln_size, 1,2,3,4,7,8,9,10,11,12,13,14,15
15	0.9576	0.9440	15.0002	0.18649	0.00405	ln_size, 1,2,3,4,5,6,8,9,10,11,12,13,14,15
15	0.9575	0.9440	15.0545	0.18671	0.00406	ln_size, 1,2,3,4,5,6,7,8,9,10,11,13,14,15
15	0.9574	0.9438	15.2131	0.18736	0.00407	ln_size, 1,2,3,4,6,7,8,9,10,11,12,13,14,15
15	0.9571	0.9434	15.5469	0.18871	0.00410	ln_size, 1,2,3,4,5,6,7,8,9,10,11,12,14,15
15	0.9565	0.9426	16.1558	0.19118	0.00416	ln_size, 1,2,3,4,5,6,7,8,10,11,12,13,14,15
15	0.9561	0.9421	16.6162	0.19304	0.00420	ln_size, 1,2,3,4,5,7,8,9,10,11,12,13,14,15
15	0.9557	0.9415	17.0482	0.19480	0.00423	ln_size, 1,3,4,5,6,7,8,9,10,11,12,13,14,15
15	0.9555	0.9413	17.2158	0.19548	0.00425	ln_size, 1,2,3,4,5,6,7,8,9,10,11,12,13,15
15	0.9550	0.9406	17.7830	0.19778	0.00430	ln_size, 1,2,3,4,5,6,7,8,9,10,12,13,14,15
15	0.9533	0.9384	19.5987	0.20514	0.00446	ln_size, 1,2,3,4,5,6,7,8,9,10,11,12,13,14
15	0.9531	0.9381	19.8358	0.20610	0.00448	ln_size, 1,2,4,5,6,7,8,9,10,11,12,13,14,15
15	0.9514	0.9358	21.7406	0.21382	0.00465	ln_size, 1,2,3,4,5,6,7,9,10,11,12,13,14,15
15	0.9507	0.9350	22.4465	0.21668	0.00471	ln_size, 1,2,3,4,5,6,7,8,9,11,12,13,14,15
15	0.9498	0.9338	23.4007	0.22055	0.00479	ln_size, 1,2,3,5,6,7,8,9,10,11,12,13,14,15
15	0.9497	0.9337	23.5187	0.22103	0.00480	ln_size, 2,3,4,5,6,7,8,9,10,11,12,13,14,15
15	0.5726	0.4363	432.313	1.87836	0.04083	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
16	0.9576	0.9428	17.0000	0.19055	0.00423	ln_size, 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15

2.3.10. Final Optimal Reduced Regression Model

Among the many solutions provided by forward selection, backward elimination, stepwise selection, Maximum R^2 improvement, and Minimum R^2 improvement, the optimal reduced model must be selected so that it can provide the managers with

reasonable suggestions. The selected reduced models must provide enough variations, in other words, the information loss must be at the most acceptable level. The selected reduced models must only contain most significant variables. By comparing all the results, the reduced model with 9 explanatory variables: ln_size, RELY, CPLX, TIME, ACAP, PCAP, VEXP, TOOL and SCED are chosen because the model's R-Square = 0.9509 >= 0.9500 is the highest in this level, the model's F-test is significant (p value <0.0001), all the p-values of the estimate parameters are significant at the level of 0.05. Table 2-18 shows the final regression model of all procedures.

Table 2-18. Optimal Reduced Regression Model

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	9	196.43079	21.82564	114.00	<.0001
Error	53	10.14727	0.19146		
Corrected Total	62	206.57806			

R-Square = 0.9509 and C(p) = 10.2535

Variable	Parameter Estimate	Standard Error	Type II SS	F Value	Pr > F
Intercept	-15.81859	1.24040	31.13756	162.63	<.0001
ln_size	1.19789	0.04591	130.32332	680.69	<.0001
RELY	1.73782	0.47279	2.58673	13.51	0.0006
CPLX	0.97757	0.41490	1.06289	5.55	0.0222
TIME	1.82230	0.51142	2.43086	12.70	0.0008
ACAP	2.39591	0.59311	3.12418	16.32	0.0002
PCAP	1.62295	0.48435	2.14962	11.23	0.0015
VEXP	3.73305	0.72238	5.11292	26.71	<.0001
TOOL	1.81828	0.77713	1.04811	5.47	0.0231
SCED	2.47115	0.78480	1.89824	9.91	0.0027

The final optimal regression model with only 9 explanatory variables of all procedures shows that its R-Square = 0.9509, which is very close to that of the full regression model with all 16 explanatory variables (0.9576). The most important factors in this model in descendent order are: ln_size, VEXP, ACAP, RELY, TIME, PCAP, SCED, TOOL and CPLX. The model can be expressed in Equation (2-32) and Equation (2-33):

$$Ln_Effort = -15.819 + 1.198 * ln_size + 1.738 * RELY + 0.978 * CPLX$$

$$+ 1.822*TIME + 2.395*ACAP + 1.623*PCAP$$
$$+ 3.733*VEXP + 1.818*TOOL + 2.471*SCED \quad (2\text{-}32)$$

that is,

$$Effort = (size)^{1.198} * \exp(-15.819 + 1.738*RELY + 0.978*CPLX$$
$$+ 1.822*TIME + 2.395*ACAP + 1.623*PCAP$$
$$+ 3.733*VEXP + 1.818*TOOL + 2.471*SCED) \quad (2\text{-}33)$$

The ln(Effort) vs. ln(size) , the estimated effort vs. actual effort of the final optimal reduced model and the full regression model are plotted in Figure 2-1 through Figure 2-4. From these plots, it is clear that the ln_Effort vs. ln_size are linear relations, it can be seen that all the slope values are fixed for the same model for ln(Effort) vs. ln(size) plots, the intercepts of the cluster of these linear relations are determined by their eight explanatory variables (cost drivers). The performance of the reduced model is very close to that of the full regression model.

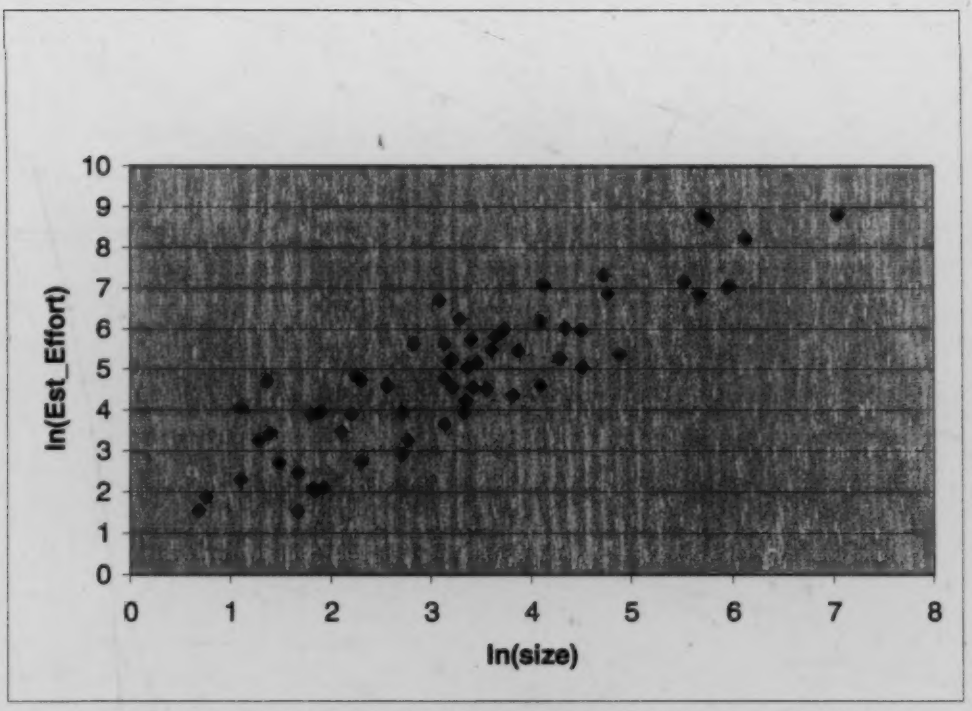

Figure 2-1. ln_Effort vs. ln_size Plot of the Reduced Multiple Regression Model with 9 Explanatory Variables

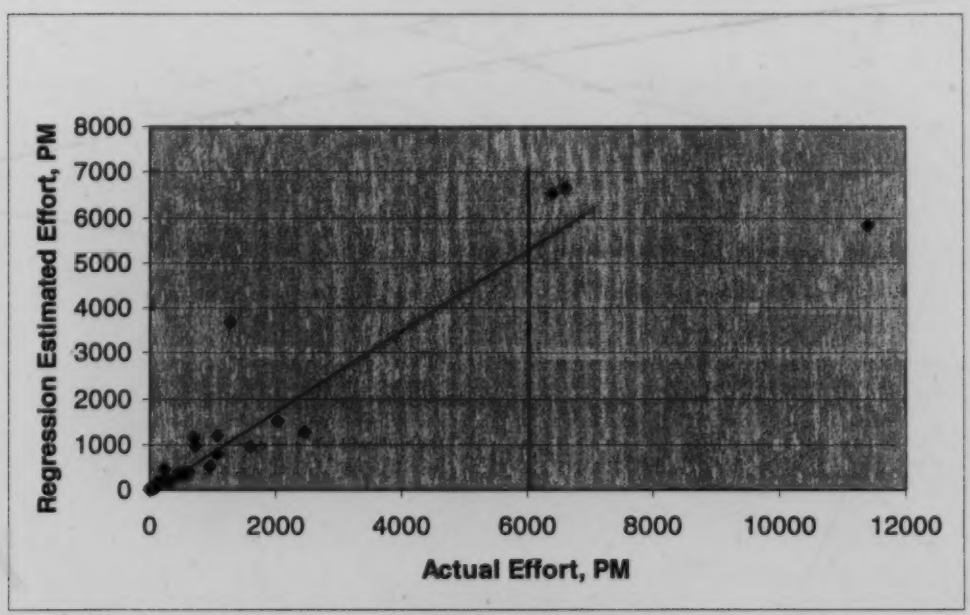

Figure 2-2. Regression Estimated Effort vs. Actual Effort by the Reduced Multiple Regression Model with 9 Explanatory Variables

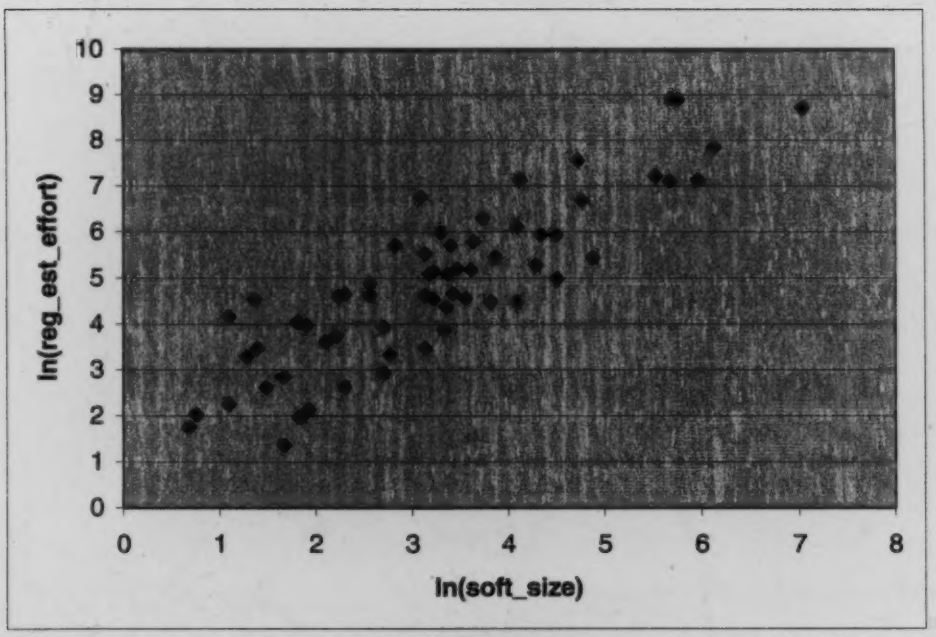

Figure 2-3. ln_Effort vs. ln_size Plot of theFull Multiple Regression Model with All 16 Explanatory Variables

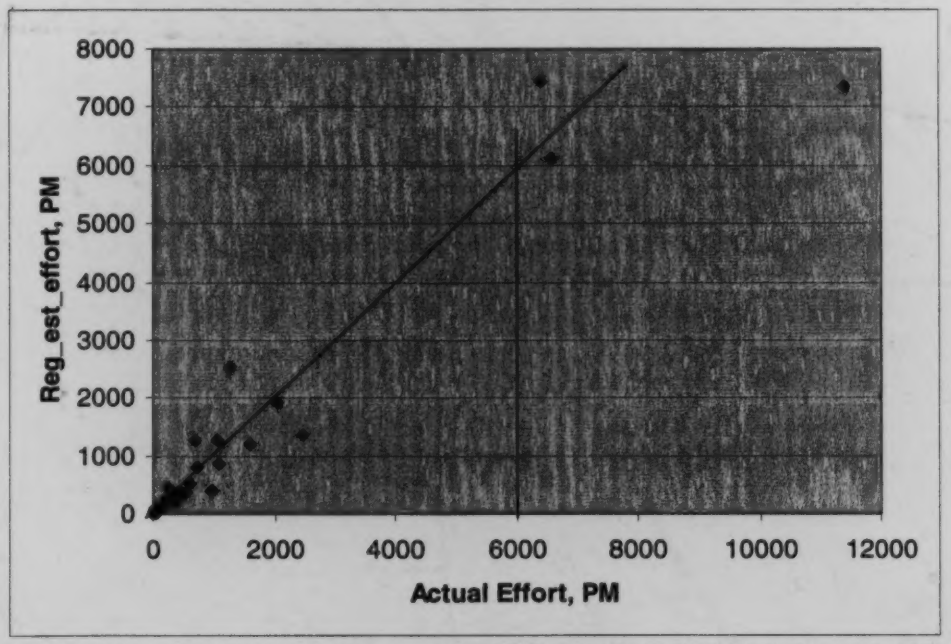

Figure 2-4. Regression Estimated Effort vs. Actual Effort by the Full Multiple Regression Model with All 16 Explanatory Variables

2.4 Summary

The final optimal reduced regression model with only 9 explanatory variables of all procedures shows that its R-Square equals 0.9509, which is very close to that of the full regression model with all 16 explanatory variables (R-Square equals 0.9576). The most important 9 factors in this model in decreasing order are: ln_size, VEXP, ACAP, RELY, TIME, PCAP, SCED, TOOL and CPLX.

Since ln_Effort vs. ln_size is linear, it can be concluded that all the slope values are fixed for the same model for ln(Effort) vs. ln(size) plots, the intercepts of the cluster of lines are determined by their eight explanatory variables.

The performance of the obtained optimal reduced model is close to that of the full regression model.

There are several potential outliers for both the full regression model and the reduced models, but their effects are not significant and thus can be tolerated.

2.5 Bibliography

[1]. B. W. Boehm, Software Engineering Economics, Prentice, 1981.

[2]. J.D. Jobson, Applied Multivariate Data Analysis, Volume 1: Regression and Experimental Design, Springer, New York, 1991.

[3]. J. Drew et al. *Case Study: factors for early prediction of software development success,* Information and Software Technology, 44 (2002) 53-62.

[4]. B. A. Kitchenham, *The question odd scale economies in software—why cannot researchers agree?* Information and Software Technology, 44 (2002) 13-24.

[5]. A. Heiat, *Comparison of artificial neural network and regression models for estimating software development effort.* Information and Software Technology,44 (2002) 911-922.

[6]. T.P. Ryan, Modern Regression Methods, Join Wiley & Sons, Inc. New York, 1997.

[7]. H. Liu and H. Motoda, Feature Selection for Knowledge and Data Mining, Kluwer Academic Publishers, 1998.

[8]. K. Cios, W. Pedrycz and R. Swinarski, Data Mining Methods for Knowledge Discovery, Kluwer Academic Publishers, 1998.

[9]. H. Liu and H. Motoda, Instance Selection and Construction for Data Mining, Kluwer Academic Publishers, 2001.

[10]. A. J. Miller, Subset Selection in Regression, Chapman & Hall, 1990

Chapter 3 Using Neural Networks for Effort Estimation Modelling

3.1 Introduction

In software engineering, the Constructive Cost Model (COCOMO) described by Barry Boehm [1] for the software effort estimation is one of the most popular tools for project management. The COCOMO '81 intermediate model makes its estimates of the required effort (measured in Person-Months – PM) based primarily on the estimate of the project's software size (as measured in thousands of SLOC, KSLOC):

$$Effort = A * EAF * (SIZE)^B \qquad (3\text{-}1)$$

Where

EAF Is the Effort Adjustment Factor derived from the 15 cost drivers

A, B Are constants derived from the development modes.

As it could be referred to Appendix 1, there are 63 samples for the COCOMO dataset, and in each sample, there are 17 dimensions: 15 cost drivers (see Appendix 2) that determines the effort adjustment factor (*EAF*), plus the project software size and the actual effort. The application type and development mode determines the values of the coefficients *A* and *B*. The theoretical *EAF* is the product of the 15 cost drivers, but the actual target *EAF* is calculated by equation (1), using the actual software size and actual efforts, and the corresponding *A, B*. *A* and *B* are derived from three development modes: organic, semi-detached, and embedded. Organic mode is used to calculate the effort where the project constraints upon development are mild, in addition, the given project has been predated by a number of similar projects, that could assist in defining the agenda of development, in the case, *A*=3.2, *B*=1.05. Embedded mode is used for a project that

has very tightly defined constraints, the project as a whole can not rely on the previous projects completed, in such case, A=2.8, B=1.20; Semi-detached mode is used for a project where the constraints for the project are greater than the organic mode, but there still remains some flexibility, the project may only be pre-dated by a few similar projects, for such situations, A=3.0, B=1.12.

In order to investigate the local relationships between the 15 cost drivers and the corresponding actual *EAF* of the 63 samples in Appendix 1, the full connected neural networks with back-propagation (BP) training and take-one-out cross validation technique is used to predict the effort adjustment factor (*EAF*) — the only output unit of the networks. To see how well the neural networks simulated *EAF* match the actual targets, the estimated efforts are calculated according to Equation (3-1) based on the estimated *EAF* matrix. The number of hidden nodes within the only hidden layer will be determined experimentally. The goal is to accurately estimate the software efforts using simple and efficient neural networks instead of COCOMO stepwise regression.

3.2 Methodology

When the neural networks are built initially, the randomly generated weights and bias ranging from –1 to 1 in the hidden layer and the output layer are added to the network. Then the networks begin to be trained and the learning process works offline. The 15 cost drivers and 1 output variable from Appendix 1 are rescaled or normalized before they are read to the neural networks, in addition, an extra unit input (value =1.0) is added as the first dimension of the input space in each sample, in other words, there are totally 16

input units, and one output unit for the networks. The simulation outputs are rescaled to real outputs before they are displayed and written to the output files.

The extreme form of n-fold cross validation technique is also called the take-one-out technique, with which could take one out of n (here n=63) samples as the testing set, while the rest corresponding n-1 (here, n-1=62) samples as the training set, therefore, there are 63 testing sets, and 63 training sets for the COCOMO dataset. For each pair of training set and testing set, there is the corresponding root mean squared error (RMSE) between the target and the neural networks output. RMSE can be calculated using equation (3-2):

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(T\arg et - NNOutput)_i^2}{N}} \qquad (3\text{-}2)$$

Where N=63, target is the actual *EAF* in Appendix 1, NNOutput is output of *EAF* from the neural networks with back-propagation training and take-one-out cross validation.

This thesis will compare the performances of the neural networks with three types of activation functions used in the output layer and the hidden layer: the unipolar sigmoid function (Equation 3-3), bipolar (Equation 3-4) and linear function:

$$f(x) = \frac{1.0}{[1.0 + \exp(-x \times \lambda)]} \qquad (3\text{-}3)$$

and bipolar sigmoid functions:

$$f(x) = \frac{2.0}{[1.0 + \exp(-x \times \lambda)]} - 1.0 \qquad (3\text{-}4)$$

Considering that there are only 63 samples in COCOMO dataset, it needs to simplify the networks architectures by limiting the number of hidden nodes fewer than or equal to 5, this thesis will focus on the neural networks using the unipolar sigmoid functions as transfer functions in the hidden layer, while using a linear transfer function $f(x) = \lambda x$ in the output layer. In the experimental measurement, the experiments under the same conditions are duplicated five times and the experiment with median RMSE will be taken as the representative one. The estimated efforts calculated by Equation (3-1) will be compared with their corresponding actual efforts so that it could show how well the estimated efforts meet the actual efforts. The accuracy is usually defined in terms of mean magnitude of relative error (MMRE) and Pred (25), which calculates the percentage of predictions that fall within 25 percent of the actual values. The lower the MMRE, the more accurate the simulation; the higher the Pred (25), the more accurate the estimation. The MMRE can be calculated by Equation (3-5):

$$\text{MMRE} = \frac{100}{N} \times \sum_{i=1}^{N} \left(\frac{|\, ActualEffort - EstimatedEffort \,|}{ActualEffort} \right)_i \qquad (3\text{-}5)$$

The effects of the factors such as the learning rate constants, the number of hidden nodes, and the momentum and λ on the RMSE and the learning process performance will be investigated.

The number of hidden nodes will be in the range from 5 to 20, and normalization of the inputs and outputs will be conducted according to Equation (3-6) and Equation (3-7):

Normalized Input = (Actual Input-Minimum Input)/(Maximum Input-Minimum Input) (3-6)

Normalized Target=(Actual Target-Minimum Target)/(Maximum Target-Minimum Target) (3-7)

In other words, it needs to make the inputs and outputs in the range of [0,1].

3.3 Experimental Results and Analysis

In order to obtain the optimal parameter combination, in the beginning, it needs to do explorative experiments under extreme parameter combination. Based on the explorative experiments' results, a series of experiments will be conducted under various parameters in a proper range. The explorative experiment is conducted under the parameters: learning rate constant is 0.015, momentum = 0.1, epoch is 10,000, and the number of hidden nodes is 20, using bipolar sigmoid functions as transfer functions in both hidden layer and output layer, the results are as Figure 3-1 and Figure 3-2. From Figure 3-1, it can be seen that after 3,000 epochs training, the RMSE is very low (<0.02), so it could be assumed that the network be well trained at 3,000 cycles. From Figure 3-2, it is clear that after 10,000 epochs training, the relation of NN-simulation outputs VS actual outputs are on the linear function f(x) = x, which means that the simulation EAF matrix and targets (actual EAF matrix) are perfectly fit.

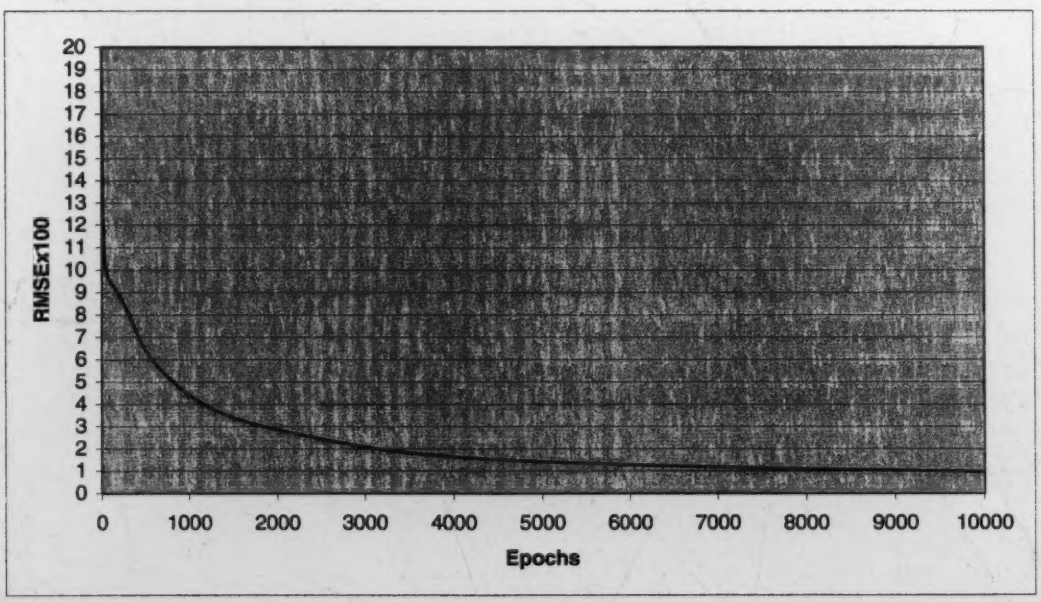

Figure 3-1. Learning Process with Bipolar Sigmoid Functions as Transfer Functions at LR=0.015

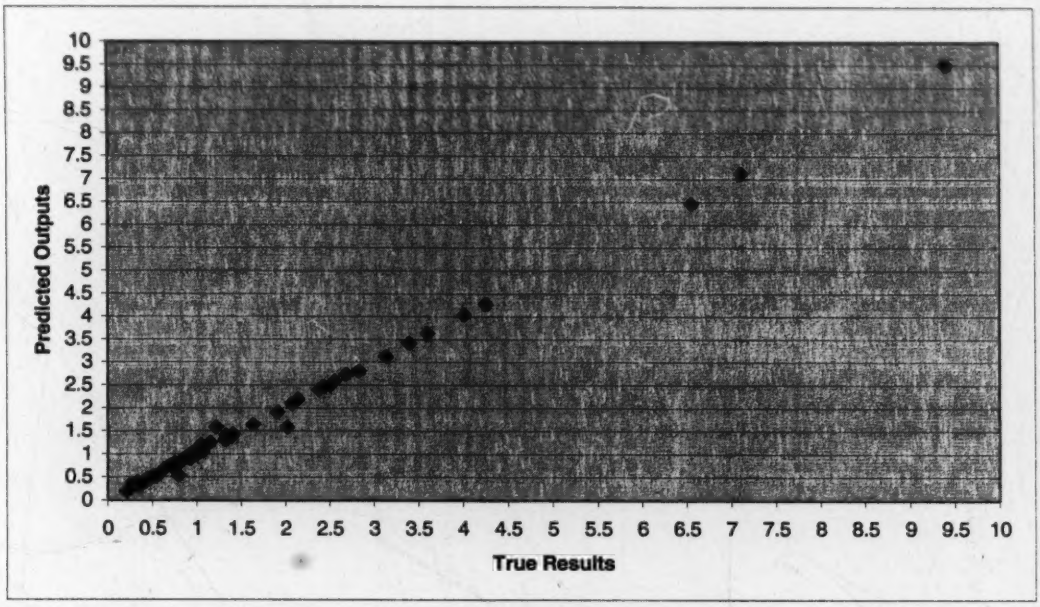

Figure 3-2. NN-Predicted EAF VS Actual EAF with Bipolar Sigmoid Functions as Transfer Functions after Epoch = 10000

Based on the explorative experiments, the experiments fix the parameter combination: epoch = 3000, momentum = 0.1.

The typical RMSE VS learning processes and the effort estimation results for the neural networks with bipolar/unipolar sigmoid functions as transfer functions in hidden layer and output layer are plotted as Figure 3-3 through Figure 3-6.

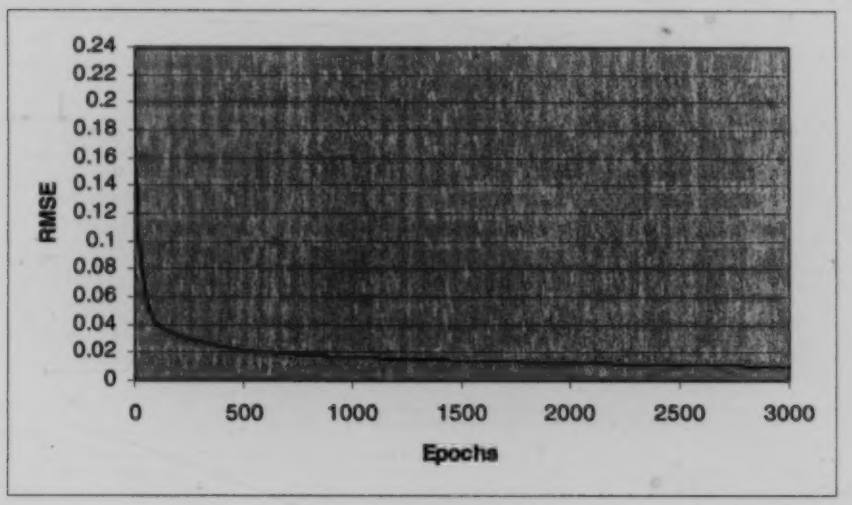

Figure 3-3. Learning Process with Bipolar Sigmoid Functions as Transfer Functions (Number of Hidden Node = 5)

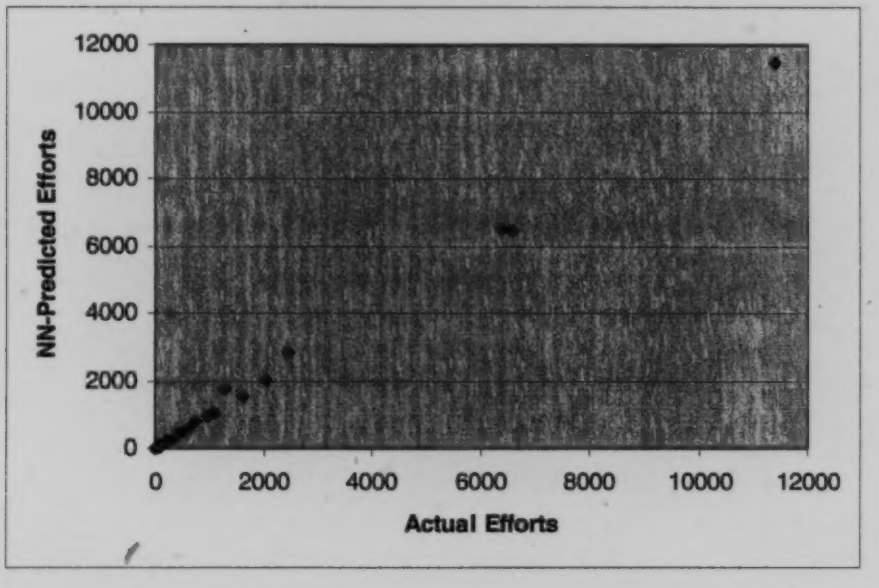

Figure 3-4. NN-Predicted Efforts VS Actual Efforts with Bipolar Sigmoid Functions as Transfer Functions (Number of Hidden Node =5)

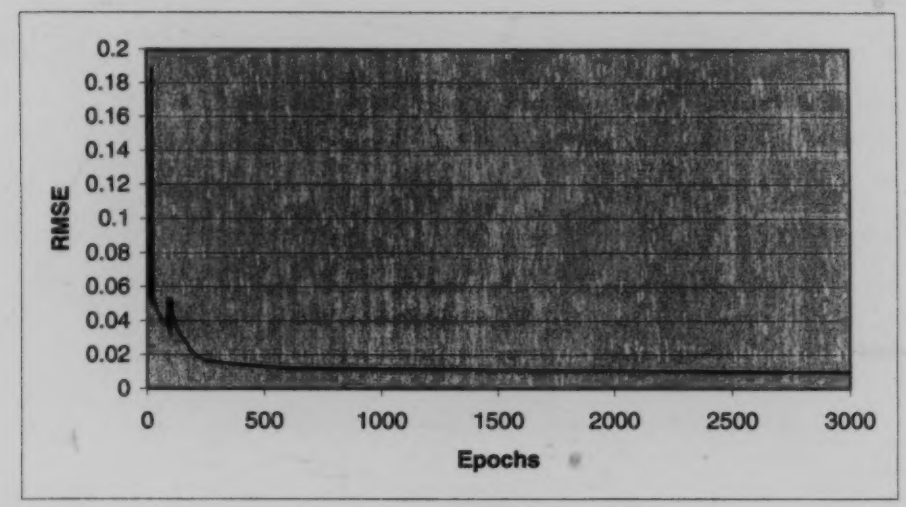

Figure 3-5. Learning Process with Unipolar Sigmoid Functions as Transfer Functions (Number of Hidden Node =5)

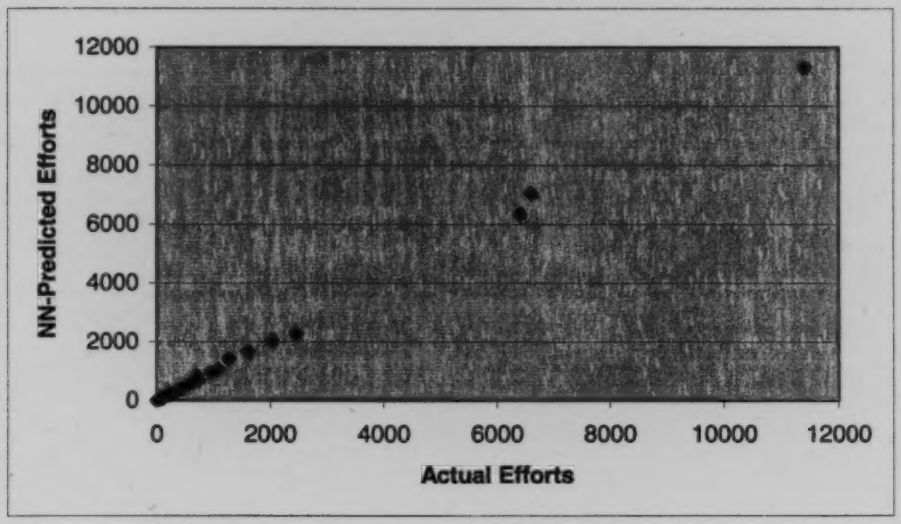

Figure 3-6. NN-Predicted Efforts VS Actual Efforts with Unipolar Sigmoid Functions as Transfer Functions (Number of Hidden Node = 5)

From Figure 3-3 through Figure 3-6, it can be seen that the simulations are well fit for the networks with bipolar or unipolar sigmoid as transfer functions in hidden layer and

output layer, the RMSE for EAF prediction, MMRE and Pred (25) for effort estimation are listed in Table 3-1.

Table 3-1. Comparison Performance of Networks with Bipolar and Unipolar Sigmoid as Transfer Functions

Transfer Type / Error Type	Bipolar-Sigmoid (1)*	Bipolar Sigmoid (2)*	Unipolar-Sigmoid (1)*	Unipolar Sigmoid (2)*
MMRE, %	1.53	7.82	3.65	7.7
Pred (25), %	98.41	90.48	93.65	93.65
RMSE	0.0079	0.0105	0.0082	0.0097

Note: (1)* represents the networks with 20 hidden nodes, (2)* is the ones with 5 hidden nodes.

Table 3-1 shows that the performance for the networks with bipolar and unipolar sigmoid as transfer functions are excellent, the difference of both networks with 5 hidden nodes and 20 hidden nodes is little, so it can be concluded that the networks with both bipolar and unipolar sigmoid as transfer functions, when suited with appropriate parameters and normalization methods, can be a very efficient effort estimator. Networks architecture simplification and optimization heavily depends on experiments.

3.3.1 Using Linear Transfer Function in Output Layer and Unipolar Sigmoid Transfer Function in Hidden Layer

Based on the results obtained in Table 3-1, this thesis tries to build the neural networks by limiting the number of hidden nodes, and using the unipolar sigmoid functions as

transfer functions in the hidden layer, while using a linear transfer function in the output layer.

3.3.1.1 Effects of Learning Rate Constants and Number of Hidden Nodes

Based on the previous results, the following experiments fix the parameter combination: epoch = 3000, momentum = 0.1, and investigate the effects of learning rates and the number of hidden nodes on RMSE, the experimental results are listed in Table 3-2. From Table 3-2, it can be seen that by limiting the number of the hidden nodes fewer than or of 5, the performance of networks is very sensitive to the learning rate constants, which are also limited in the lower level. With increasing hidden nodes number, the learning capability tends to be higher, when the number of hidden node equals 5 and the learning rate equals 0.02, the optimal RMSE converges to 0.107. However, when trying to increasing the learning rates higher than 0.1, the networks lose their learning abilities.

Table 3-2. Effects of Learning Rates and Hidden Nodes on RMSE

Using Linear Transfer Function in Output Layer

RMSE	Hidden Layer Nodes Number			
LR	2	3	4	5
0.005	1.705	0.429	0.1476	0.4435
0.010	1.711	1.714	0.4997	0.4411
0.015	0.4301	0.432	0.4346	0.1478
0.020	1.7069	0.431	0.4403	0.1071
0.025	1.7163	0.452	0.1428	0.4434
0.030	1.7159	0.439	0.1413	0.2283
0.035	1.7157	0.441	0.4467	0.1884
0.040	1.7162	0.444	0.4692	0.1871
0.10	0.454	1.127	0.5807	0.2108
0.15	1.897	1.972	2.0278	2.0278

Table 3-3. The Effects of Number of Hidden Node on RMS at Constant LR

Hidden Nodes Number	RMSE (LR=0.02)	Learning Process Performance	RMSE (LR=0.10)	Learning Process Performance
5	0.107	Stable	0.222	Unstable
8	0.0916	Unstable	0.818	Unstable
10	0.098	Unstable	2.0997	Not Convergent
12	0.0713	Unstable	2.0996	Not Convergent
15	0.0789	Stable	2.350	Not Convergent
20	0.0726	Stable	9.0017	Not Convergent

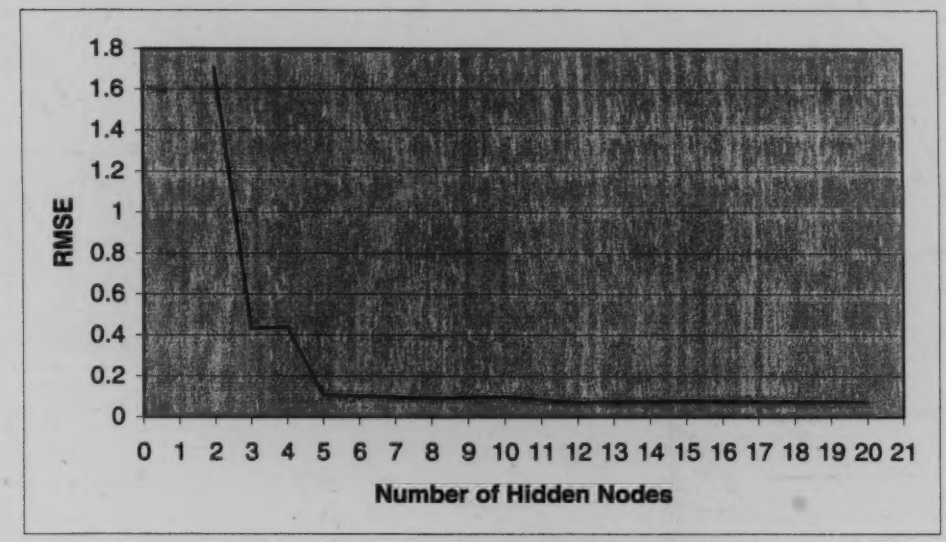

Figure 3-7. RMSE VS Number of Hidden Nodes at LR=0.02

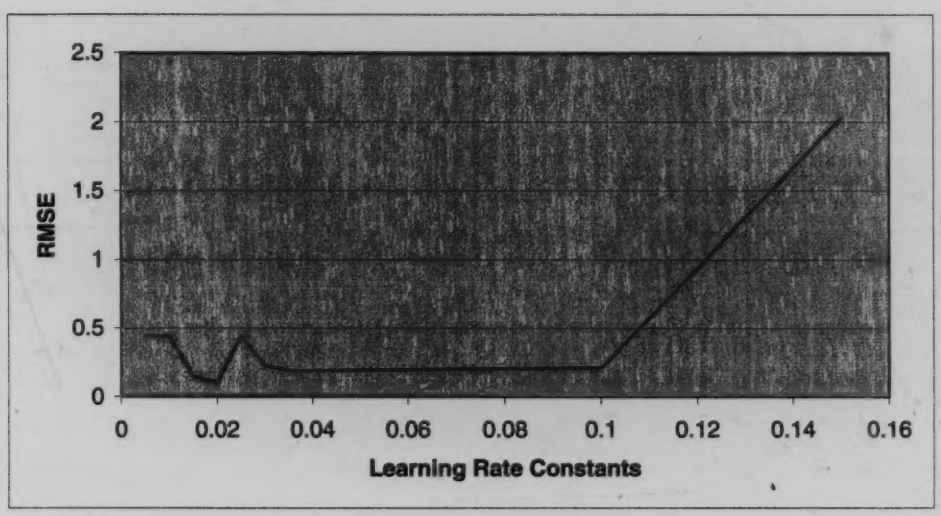

Figure 3-8. RMSE VS Learning Rate Constants at Number of Hidden Nodes =5

3.3.1.2 Effects of Momentum

Since the performance of the networks is very sensitive to the learning rate constants, it needs to look for comprehensive means of momentum, λ and learning rate constants to

control the learning processes. If the learning rate is too high, then learning can become unstable, that means the RMSE in learning curve oscillates, the net oscillates back and forth across the error minimum. One way to overcome this limitation is to alter the training rule from pure gradient descent to include a term that includes a proportion of last weight change [2,3,4]. The new rule is:

$$\Delta w_i^j{}_{(n)} = \eta \times f'(a^j)(t^j - y^j) \times X_i^j + \text{momentum} \times \Delta w_i^j{}_{(n-1)} \qquad (3\text{-}8)$$

Where η is the learning rate, t is target, y is the neuron output, X is the neuron inputs, $f'(a^j)$ is the gradient descent of the transfer function, $\Delta w_i^j{}_{(n)}$ is the current weight change, $\Delta w_i^j{}_{(n-1)}$ is the last weight change. Thus, if the previous weight change was large, the new weight change would be large too. That is, the weight change carries along some momentum to the next iteration, this has a tendency to smooth out small fluctuations in the error-weight space (it is a low-pass filter).

The fixed default parameter combination is: the number of hidden nodes is 5, epoch =3000, Learning rate =0.02. Investigate the effects of momentum on the networks RMSE. The experimental results are reported in Table 3-4.

Table 3-4. Effects of Momentum on the Networks Performance

Using Linear Transfer Function in Output Layer at LR=0.02

Momentum Values	RMSE (λ =0.5)	Learning Process Performance	RMSE (λ =1.0)	Learning Process Performance
0.0	0.431	Stable	0.369	Stable
0.1	0.110	Stable	0.107	Stable
0.2	0.168	Stable	0.173	Unstable
0.3	0.429	Unstable	0.207	Unstable

Table 3-4 shows that momentum =0.1 and λ=1.0 will produce optimal performance: RMSE =0.107 and the learning process is stable. The momentum parameter determines how much of the previous weight change will be retained in the present weight change computation. Thus, weight changes can build up momentum over time if they all head in the same direction, which can speed up learning.

3.3.1.3 Effects of λ and Learning Rate Combination

With the increasing of λ, the RMSE value convergence would be speed up at appropriate learning rates and momentums. Table 3-5 represents the effects of λ on RMSE, which shows that at higher level of learning rate (LR=0.10), it may cause the side effects that do not help speed up the RMSE convergence process, furthermore, it would cause the network to lose their learning abilities.

Table 3-5 The Effects of λ on RMSE at Constant LR

λ	RMSE (LR=0.02)	Learning Process Performance	RMSE (LR=0.10)	Learning Process Performance
0.1	0.428	Stable	0.320	Unstable
0.2	0.179	Stable	0.125	Unstable
0.3	0.133	Stable	0.197	Unstable
0.4	0.168	Stable	0.251	Unstable
0.5	0.110	Stable	0.505	Unstable
0.6	0.178	Stable	1.837	Not Convergent
1.0	0.107	Stable	1.925	Not Convergent

3.3.2 Simulation Using Categorical Data instead of Numerical Data

In COCOMO'81, the cost drivers are ranked categorically to "very low", "low", "nominal", "high", "very high", and "extra high" and their values are assigned accordingly. From Boehm's book [1], Table 3-6 reports the corresponding relationships between the categorical and numerical data of the cost drivers. The numbers inside the parentheses are the corresponding categorical numbers shown in Appendix 3.

Table 3-6 Relationships between Numerical and Categorical Input Variables for Appendix 3

Input Units and Explanations	Ratings					
Cost Drivers	Very Low	Low	No-minal	High	Very High	Extra High
Product Attributes						
RELY: Required Software Reliability	0.75(1)	0.88(2)	1.00(3)	1.15(4)	1.40(5)	
DATA: Data Base Size		0.94(1)	1.00(2)	1.08(3)	1.16(4)	
CPLX: Product Complexity	0.70(1)	0.85(2)	1.00(3)	1.15(4)	1.30(5)	1.65(6)
Computer Attributes						
TIME Execution Time Constraint			1.00(1)	1.11(2)	1.30(3)	1.66(4)
STOR: Main Storage Constraint			1.00(1)	1.06(2)	1.21(3)	1.56(4)
VIRT: Virtual Machine Volatility		0.87(1)	1.00(2)	1.15(3)	1.30(4)	
TURN: Computer Turnaround Time		0.87(1)	1.00(2)	1.07(3)	1.15(4)	
Personnel Attributes						
ACAP: Analyst Capability	1.46(1)	1.19(2)	1.00(3)	0.86(4)	0.71(5)	
AEXP: Applications Experience	1.29(1)	1.13(2)	1.00(3)	0.91(4)	0.82(5)	
PCAP: Programmer Capability	1.42(1)	1.17(2)	1.00(3)	0.86(4)	0.70(5)	
VEXP: Virtual Machine Experience	1.21(1)	1.10(2)	1.00(3)	0.90(4)		
LEXP: Program Language Experience	1.14(1)	1.07(2)	1.00(3)	0.95(4)		
Project Attributes						
MODP: Modern Programming Practices	1.24(1)	1.10(2)	1.00(3)	0.91(4)	0.82(5)	
TOOL: Use of Software Tools	1.24(1)	1.10(2)	1.00(3)	0.91(4)	0.83(5)	
SCED: Development Schedule	1.23(1)	1.08(2)	1.00(3)	1.04(4)	1.10(5)	

Based on the previous experimental results obtained from the networks using numerical input units and unipolar sigmoid function in hidden layer and linear function in output layer as transfer functions, the optimal parameters combination is: number of hidden

nodes =5, learning rate =0.020, epoch=3000, λ=1.0, and momentum=0.1. In this section, the optimal parameters combination is adopted except that the categorical input variables shown in Appendix 3 are used to replace the numerical input variables shown in Appendix 1. The experimental results are plotted as Figure 3-9 and Figure 3-10. From Figure 3-9, it can be seen that the RMSE of the learning process converges very fast and as low as 0.1 in about epoch =250, and the performance is very stable. The performance is much better than that of the same networks with the same structure but using numerical data as inputs.

The calculated RMSE = 0.0847 for EAF prediction, and MMRE = 4.95%, Pred(25) = 92.06% for the effort estimation. The simulation results are very close to the targets.

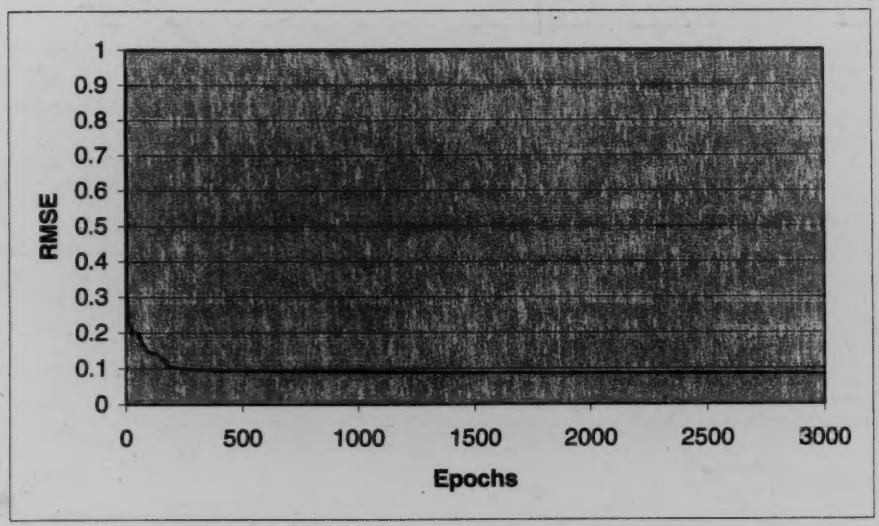

Figure 3-9 Learning Process of Categorical Data Processing Networks with Linear Function as Output Layer Transfer Function at LR=0.02

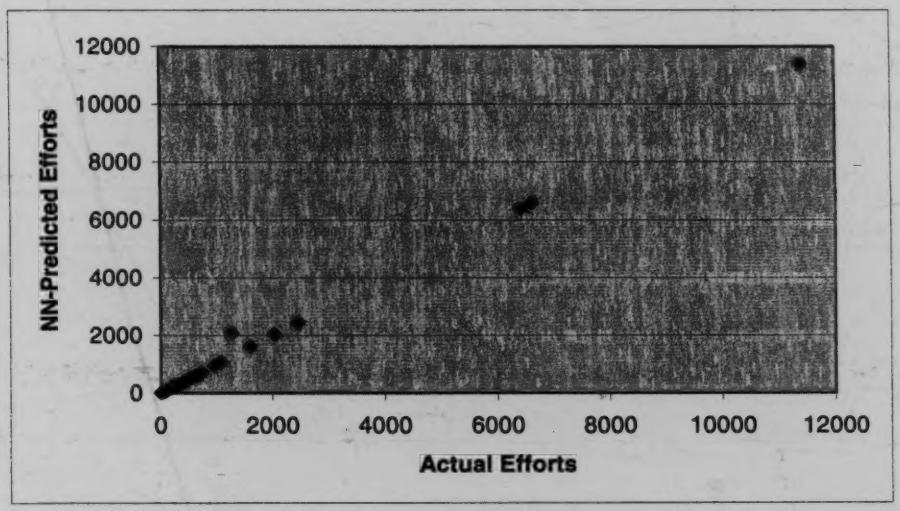

Figure 3-10 NN-Predicted Efforts VS Actual Efforts of Categorical Data Processing Networks with Linear Function as Output Layer Transfer Function at LR=0.02

3.3.3 Comparisons of Various Simulations with Optimal Parameters

For the convenience to compare the optimal performance under optimal parameter combination of the different architectures of networks, the results of predicting EAF and effort estimation are listed in Table 3-7. In the networks with bipolar sigmoid functions as transfer functions in both hidden layer and output layer, the optimal parameter combination is: learning rate = 0.05, epoch = 3000, momentum = 0.1, λ =1.0; for networks with the unipolar sigmoid as transfer functions in both the hidden layer and output layer, the optimal parameter combination is: learning rate =0.95, epoch = 3000, momentum = 0.1, λ =1.0; for the networks with unipolar sigmoid as transfer function in hidden layer and the linear function $f(x) = \lambda x$ as the transfer function in output layer, the optimal parameter combination is: learning rate =0.02, epoch = 3000, momentum = 0.1, λ

=1.0, number of hidden nodes = 5; please note that (1)* represents the number of hidden node =20, (2)* represents number of hidden node =5.

Table 3-7 Comparison of Performance of Networks with Various Transfer Functions

Transfer Type / Error Type	Bipolar-Sigmoid (1)*	Bipolar Sigmoid (2)*	Unipolar-Sigmoid (1)*	Unipolar Sigmoid (2)*	Unipolar-Linear Function	Stepwise Regression (COCOMO)
MMRE, %	1.53	7.82	3.65	7.70	16.23	18.22
Pred(25),%	98.41	90.48	93.65	93.65	77.78	74.68
RMSE	0.0079	0.0105	0.0082	0.00968	0.107	0.174

Note: (1)* represents the networks with 20 hidden nodes, (2)* is the ones with 5 hidden nodes.

3.3 Summary

Experimental results showed that the performance of the neural networks with unipolar sigmoid in hidden layer and linear function in output layer as transfer function is affected by the investigated factors: learning rate constants, hidden nodes number, momentum, lambda values. One of the three factors: learning rate constants, momentum and lambda values can be helpful to speed up the learning processes and the RMSE convergence, but it is advisable to just control one of them and let other two factors be in lower or normal levels.

When using the bipolar sigmoid function as transfer function, the optimal parameter combinations are: epoch = 3000, momentum = 0.1, hidden nodes number = 20, learning rates = 0.05, λ = 1.0. The optimal RMSE between the predicted EAF and the target EAF is as low as 0.0080, the effort estimation MMRE is as low as 1.53%, and the Pred (25) is as high as 98.41%; For the neural networks with the unipolar sigmoid function as transfer

function in both hidden layer and output layer, the optimal parameter combinations are: epoch = 3000, momentum = 0.1, hidden nodes number = 20, learning rates = 0.95, λ = 1.0, The optimal RMSE between the predicted EAF and the target EAF is as low as 0.00816, the effort estimation MMRE is as low as 3.65%, and the Pred (25) is as high as 93.65%;. Both of the learning processes are very stable. When the number of hidden nodes is reduced to 5, the optimal RMSE between the predicted EAF and the target EAF is as low as 0.0105, the effort estimation MMRE is as low as 7.82%, and the Pred (25) is as high as 90.48% for networks with bipolar sigmoid as transfer functions. While for the networks with unipolar sigmoid as transfer functions in both layers, the optimal RMSE between the predicted EAF and the target EAF is as low as 0.00968, the effort estimation MMRE is as low as 7.70%, and the Pred (25) is as high as 93.65%. For the networks with unipolar sigmoid in hidden layer and linear function in output layer as transfer functions, the optimal parameter combination is: epoch = 3000, momentum = 0.1, hidden nodes number = 5, learning rates = 0.020, λ = 1.0. The optimal RMSE between the predicted EAF and the target EAF is as low as 0.107, the effort estimation MMRE is as low as 16.23%, and the Pred (25) is as high as 77.78%.

Based on the optimal parameters combination of the networks with unipolar sigmoid function in the hidden layer and the linear function in the output layer, use the categorical input variables instead of the continuous numerical input variables, the networks with categorical input variables outperform the ones with continuous numerical input variables.

In the future, there is a great need to build more efficient, more powerful and more simplified architectures networks to obtain more accurate effort estimation for real world projects.

3.5 Bibliography

[1] B. W. Boehm, Software Engineering Economics, Prentice, 1981.

[2] A. Heiat, *Comparison of artificial neural network and regression models for estimating software development effort.* Information and Software Technology, 44 (2002) 911-922.

[3] C. G. Looney, Pattern Recognition Using Neural Networks: theory and algorithms for engineers andscientists, Oxford University Press, NY, 1997

[4] S. Haykin, Neural Networks: A Comprehensive Foundation, Second Edition, Prentice Hall, 2001.

Chapter 4 Optimal Attribute Subset Selection Using Genetic Algorithms

4.1 Introduction

Genetic Algorithms (GAs) may deal successfully with a wide range of domains. The reasons [1, 2, 3, 4] for this success are: 1). GAs can solve hard problems quickly and reliably, 2) GAs are easy to interface with existing simulations and models, 3) GAs are extensible and 4) GAs are easy to hybridize. In other words, GAs are robust.

There are many attribute selection methods for linear models, for example, in our previous research work in Chapter 2, some linear methods such as forward selection, backward elimination, stepwise selection were successfully employed to build the optimal reduced regression models for the COCOMO dataset, however, such an optimal regression reduced model is not necessary to be optimal in the nonlinear simulation, GAs are well known for the successful applications in attribute search and optimization, it can be concluded that GAs are efficient tools to reach the goals of this thesis: finding out the globally optimal subset of 8 attributes from the 15 cost drivers in COCOMO dataset. The software size as a fixed significant factor is added to the 8 optimal attributes, thus composes the 9 dependent variables/inputs in the reduced model of regression and neural network simulation.

4.2. Methodology of the Real-Valued Genetic Algorithms

The basic genetic algorithm is as follows:

1. [Start] Generate random population of n chromosomes (suitable solutions for the problem)

2. [Fitness] Evaluate the fitness $f(x)$ of each chromosome x in the population

3. [New population] Create a new population by repeating following steps until the new population is complete

3a. [Selection] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)

3b. [Crossover] With a crossover probability cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.

3c. [Mutation] With a mutation probability mutate new offspring at each locus (position in chromosome).

3d. [Accepting] Place new offspring in the new population

4. [Replace] Use new generated population for a further run of the algorithm

5. [Test] If the end condition is satisfied, stop, and return the best solution in current population

6. [Loop] Go to step **2**

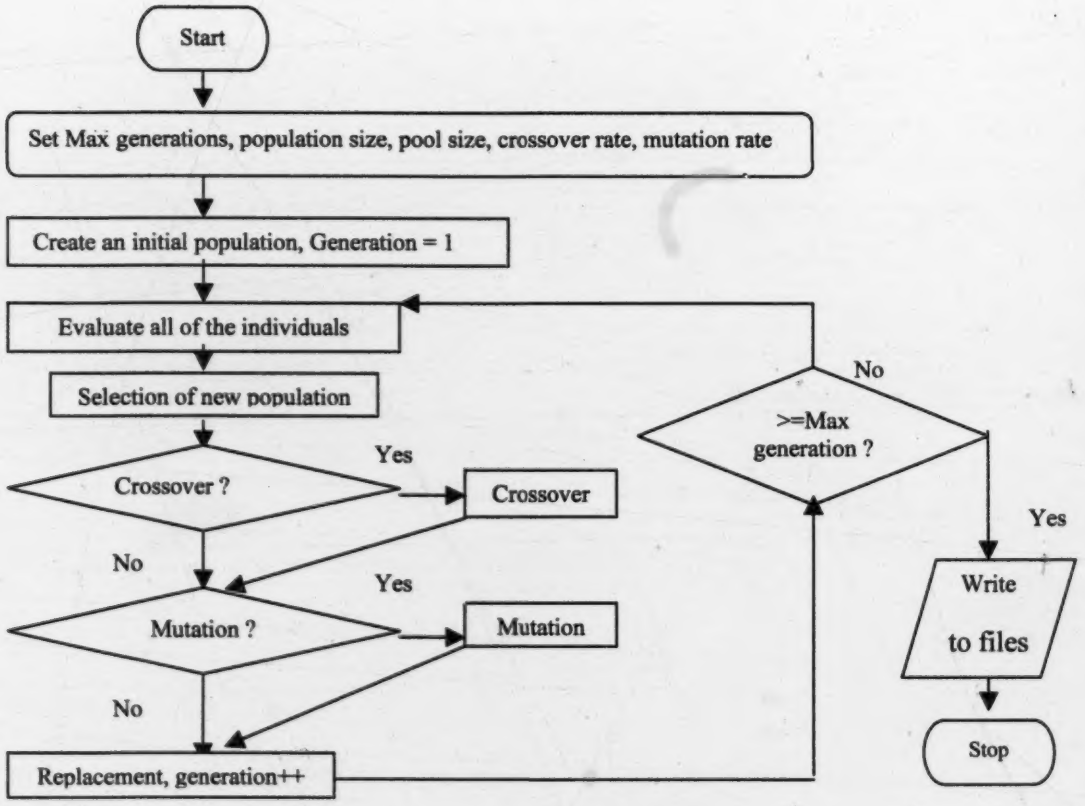

Figure 4-1 Flowchart of basic genetic algorithm

A. Initialization function: Create an initial population p of chromosomes at generation 1:

P is the population size. In the initial state, the first generation of chromosomes is randomly produced, according to the set population size, chromosome length, and the interval of each gene's value. For example, if the population size = 100, chromosome length = 10, and the interval of each gene's value is [0,1], there are 100 chromosomes in each generation, each chromosome looks like the following:

Gene Index	1	2	3	4	5	6	7	8	9	10
Gene Value	0.657	0.0184	0.524	0.835	0.133	0.907	0.012	0.627	0.860	0.837

Figure 4-2 Sample of chromosome represented in real valued coded in [0,1]

The local time is a seed for the random generator function, srand(time(NULL)); the interval of the gene's value is in [0,1], the constant RND can be defined as: #define RND ((float) rand()/((float) RAND_MAX+1)).

B. Evaluation fitness:

For each of the chromosome, the Evaluation function locates the gene's position by taking the first 8 genes whose values are the first 8 largest from the total 15 genes in the case as shown in Figure 4-3. Call the function of Find Fitness where training relative error between the target and the actual output from the neural networks or multiple linear regression is returned to calculate the fitness of the chromosome. The fitness value and the corresponding chromosome with higher fitness will be fed to the next generation selection.

Index	6	9	10	4	1	8	3	5
Value	0.907	0.860	0.835	0.837	0.657	0.627	0.524	0.133

Figure 4-3 Sample of Chromosome with Optimal Subset of 8 Variables (Gene's Indexes) from Figure 4-2

Foe example, in Figure 4-3, the corresponding variable subset = {6,9,10,4,1,8,3,5}, it is passed as pointer of array to the function of Find Fitness to calculate the training relative

error between the target and the actual output using the neural networks or multiple linear regression.

C. Selection function:

First, all the chromosomes of the previous generation are copied to the temporary generation without any operations of crossover or mutation, they compose the first part of the temporary generation. The second part of the temporary generation is produced after applying tournament selection mechanism to the previous generation.

Tournament selection works in the following way: a pair of winners is selected separately from two groups of chromosomes among the population of previous generation. Each group has n (n>=2) group members are randomly picked up from the entire previous population of candidates. From the members of each group, the chromosome with the highest fitness value is the winner. The pair of winners in each group is passed to crossover if they are selected to be parents, otherwise they are directly subjected to mutation. In the temporary generation, the tournament selection process is repeatedly applied so as to obtain a breeding population that is equal in size to the original population of candidate chromosomes. By far, in the temporary generation there are twice of population size compared with the required population size. Therefore, the half of the chromosomes with best fitness values is selected. These selected chromosomes form the new population in the next generation.

For example, suppose that there are the following chromosomes with their corresponding fitness values:

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10

Chromosome 1, supposed that its Fitness = 108.89;

B1	B2	B3	B4	B5	B6	B7	B8	B9	B10

Chromosome 2, supposed that its Fitness = 100.23;

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10

Chromosome 3, supposed that its Fitness = 98.34;

Figure 4-4 Samples of Chromosomes with Corresponding Fitness Values in a Pool

Select chromosome 1 as the best in this group in Figure 4-4.

D. Crossover function:

The simple crossover mechanism is chosen: For each pair of the chromosomes selected to be crossover, at the random position, the second part of the chromosome is exchanged to the same positions of its partner chromosome.

Selection scheme for crossover: for each pair of chromosomes, randomly produces a real number in [0,1], compare it to the given crossover rate, if it is less than the latter, then this pair of chromosomes are selected as parents for crossover operation, otherwise, pass the pair to mutation operation directly without crossover.

Suppose that there are two parent chromosomes selected crossover, A and D.

A={A1, A2, A3, A4, A5, A6, A7, A8, A9, A10}; D= {D1, D2, D3, D4, D5, D6, D7, D8, D9, D10}. Simple crossover generates a random number *n* from uniform distribution from 1 to 10 and creates two new chromosomes A' and D' as offspring. A'={A'1, A'2, A'3, A'4, A'5, A'6, A'7, A'8, A'9, A'10}; D' = {D'1, D'2, D'3, D'4, D'5, D'6, D'7, D'8, D'9, D'10}.

$$A_i' = \begin{cases} A_i, i < n \\ D_i, otherwise \end{cases} \qquad (4-1)$$

$$D_i' = \begin{cases} D_i, i < n \\ A_i, otherwise \end{cases} \qquad (4-2)$$

The simple crossover operation is shown in Figure 4-5.

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
						X			
D1	D2	D3	D4	D5	D6	D7	D8	D9	D10

In the case of n = 7, the offspring is A' and D' as following: ⇩

A1	A2	A3	A4	A5	A6	D7	D8	D9	D10
D1	D2	D3	D4	D5	D6	A7	A8	A9	A10

Figure 4-5 Simple Crossover Operation

E. Mutation function:

The simple mutation mechanism is chosen: for each gene of all chromosomes after crossover or without crossover, at any given gene, randomly produces a real number in [0,1], compare it with the given mutation rate, for example, 0.05, if the given mutation rate p_m = 0.05 is greater than the randomly produced real number *U (0,1),* the new chromosome's gene value xi' is assigned to 1.0-xi as shown in Figure 6, where in the locations of the two genes whose value is x4 and x7 the random number *U (0,1)* are supposed to produce and their *values* < the given mutation rate p_m (0.05), then their genes' values must be mutated to become (1.0-x4) and (1.0-x7), respectively. Otherwise, their genes' values keep unchanged. Suppose that X is the original chromosome, X' is the

mutated chromosome; X = {xi, i=1,...,10}; X' = {xi', i = 1,...,10}, the procedure of mutation can be indicated as Equation (4-3).

$$x_i' = \begin{cases} 1.0 - x_i, if U(0,1) < p_m \\ x_i, otherwise \end{cases} \qquad (4\text{-}3)$$

Original chromosome:

x1	x2	x3	x4	x5	x6	x7	x8	x9	x10

Mutated chromosome:

x1	x2	x3	(1.0-x4)	x5	x6	(1.0-x7)	x8	x9	x10

Figure 4-6 Mutation Operation

F. Find Fitness function:

Pass the suggested optimal subset of variables of each chromosome to the neural network to select the input nodes, the training relative error between the target and the actual output from the neural networks/or the multiple linear regression is returned to calculate the fitness value of the chromosome in the function Evaluation.

The relative training error root mean square of error *(rmse)* is defined as Equation (4-4) and the fitness is calculated using Equation (4-5):

$$rmse = \sqrt{\frac{\sum_{i=1}^{n} (target_i - output_i)^2}{n}} \qquad (4\text{-}4)$$

$$fitness = 1000.0 \times \exp(-rmse) \qquad (4\text{-}5)$$

It is important to note the distinction between the fitness and the objective scores. The objective score is the value returned by the objective function; it is the raw performance

evaluation of a chromosome. The fitness score, on the other hand, is a rating by the genetic algorithm to determine the fitness of individuals for mating. The fitness score is typically obtained by a scaling of the raw objective scores. Here *rmse* is the objective score from the objective function (we used neural networks as objective function) for each chromosome, their values are real numbers less than 1.0 in our case; the lower the value of *rmse*, the better the performance of the chromosome. But they may sometimes be too close to be easily distinguished. Equation (4-5) is proposed to map fitness score between *-rmse* with exp function because exp function is a built-in continuous function that can map any point of *-rmse* to its unique corresponding fitness value in the domain of [0,1], 1000.0 is a scaling factor to ensure that the fitness value is in significantly difference so that it is easier for the genetic algorithm to tell whether a chromosome is better than the other based on fitness comparison.

G. Replacement function:

For each new generation, the chromosomes from previous generation and current generation are mixed to form a temporary generation, pick the best population size of chromosomes to the next generation.

H. Data Structures Used in Genetic Algorithm Implementations:

Although the use of user-defined data types (or records) will produce a slightly more elegant program, these will result in a program that is harder to understand. The population is held in a series of two-dimensional arrays with each array representing one chromosome of the population, the data type is "double" and the genes' values are in

[0,1]. The optimal subset of variables are integers from 0 to 14 and also, are two-dimensional arrays with each array presenting as an optimal subset of variables for one chromosome. It is also the subset of the input variables for neural networks in FindFitness function. The fitness values of the population of the chromosomes are held in one-dimensional array and the data type is "double". For the best chromosome, use the arrays to record the chromosome, its fitness value, and its optimal subset of variables. These arrays are global for all functions in this program.

4.3 Experimental Results and Analysis

From related experience [1], combining the purpose this thesis to select 8 optimal cost drivers plus the project software size as the 9 most important attributes in the simplified model, it can narrow the experimental conditions as following:

Population Size =100; Maximum Generation = 100; Total Variables = 15; Number Of Selected Variables = 8; Crossover rate =0.75~0.95; Mutation rate =0.01~0.2.

4.3.1 Using Multiple Regression to Evaluate Fitness

From Figure 4-7 and Figure 4-8, it is clear to see that the GA's best fitness and average fitness values are increased with generations, this confirms that the real valued coded GA always keeps the best chromosomes in the generation and next generation is better that the previous one. Table 4-1 reports the results for various crossover rates (CR) and mutation rates (MR).

In Table 4-1, the optimal sunset obtained by GA are the same as the final optimal one by linear techniques in Chapter 2, The most important 9 factors in this model are: ln_size,

VEXP, ACAP, RELY, TIME, PCAP, SCED, TOOL and CPLX. It can be concluded that the optimal subset obtained by multiple linear regression in Chapter 2 is the globally optimal subset.

The interesting finding using GA and reduced regression models is that in the COCOMO dataset, personnel attributes are the most significant variables, followed by project attributes and product attributes. Computer attributes are least important in determining the software development effort. Therefore, human capabilities and project management skills are still the most significant factors in software development processes.

Table 4-1. Optimal Subset of Variables by GA Using Multiple Regression to Evaluate Fitness

Subset Label	Optimal Subset	GA Parameters	GA-Mreg RMSE*	Best Fitness	Average Fitness
Linear-search	1,3,4,8,10,11,14,15	NONE	0.174297	840.048	N/A
GA-Mreg-952	1,3,4,8,10,11,14,15	CR=0.95, MR=0.2	0.174297	840.048	830.748
GA-Mreg-951	1,3,4,8,10,11,14,15	CR=0.95, MR=0.1	0.174297	840.048	831.173
GA-Mreg-9501	1,3,4,8,10,11,14,15	CR=0.95,MR=0.01	0.174297	840.048	830.826
GA-Mreg-852	1,3,4,8,10,11,14,15	CR=0.85, MR=0.2	0.174297	840.048	832.557
GA-Mreg-851	1,3,4,8,10,11,14,15	CR=0.85, MR=0.1	0.174297	840.048	832.092
GA-Mreg-8501	1,3,4,8,10,11,14,15	CR=0.85, MR=0.01	0.174297	840.048	831.793
GA-Mreg-752	1,3,4,8,10,11,14,15	CR=0.75, MR=0.2	0.174297	840.048	835.004
GA-Mreg-751	1,3,4,8,10,11,14,15	CR=0.75, MR=0.1	0.174297	840.048	833.821
GA-Mreg-7501	1,3,4,8,10,11,14,15	CR=0.75, MR=0.01	0.174297	840.048	826.886

Each experiment of the nine different GA conditions can find the globally optimal subset, this globally optimal subset was also found by the linear regression techniques in Chapter 2; it was also found that in each level of best fitness value before reaching 840.048, the

corresponding optimal subsets have more common members with the globally optimal subset as their fitness values are closer to the 840.048.

Figure 4-9 is the reduced multiple regression model with the globally optimal subset found by GA with multiple regression as fitness function and Figure 4-10 is the full regression model, it can be seen that their performance in effort estimation is very close.

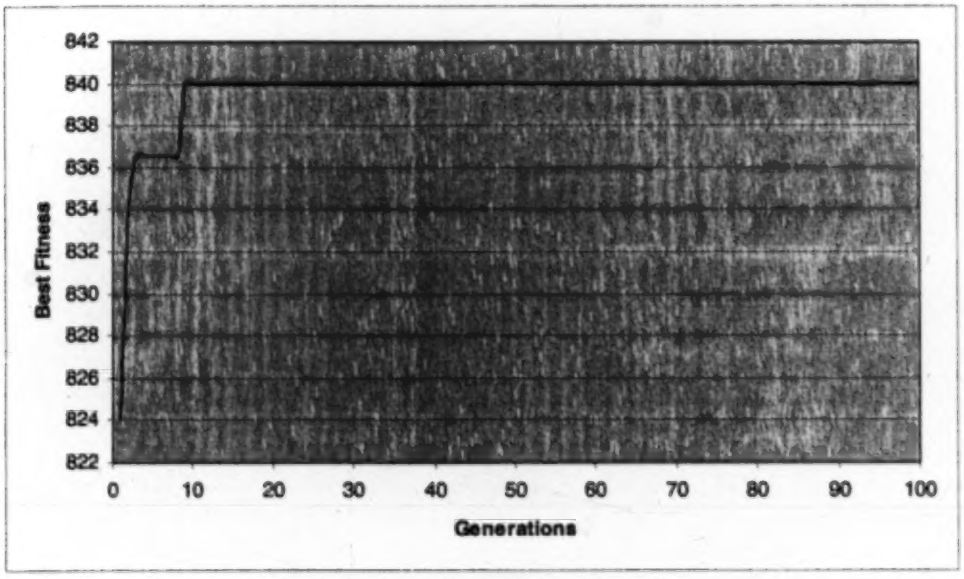

Figure 4-7 Typical Relations of Best Fitness VS Generations of GA Using Multiple Regression as Fitness Function

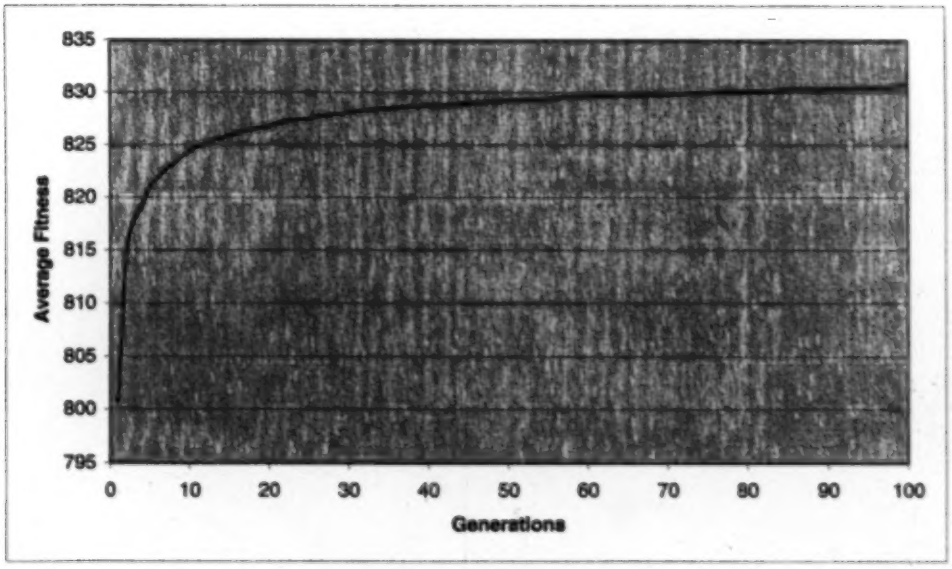

Figure 4-8 Typical Relations of Average Fitness VS Generations of GA Using Multiple Regression as Fitness Function

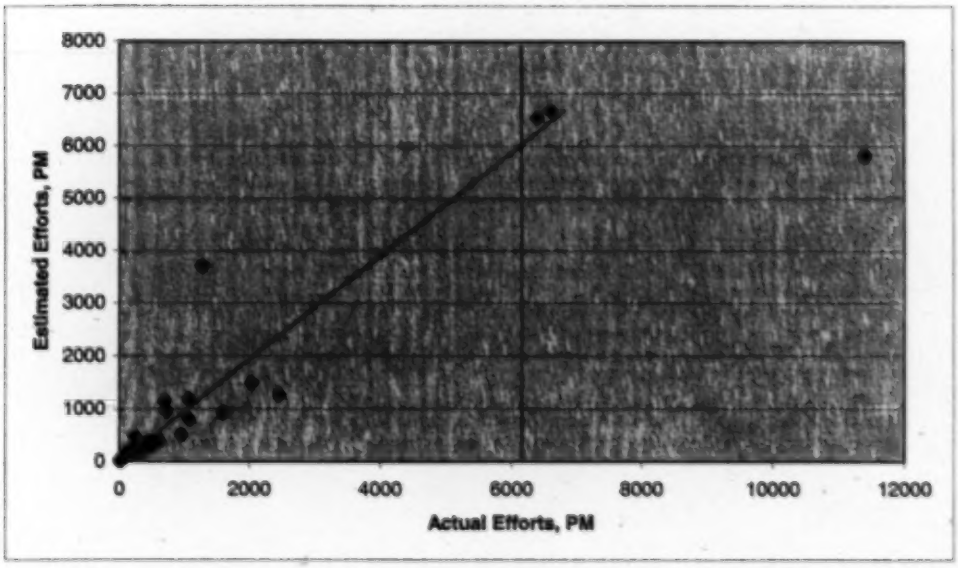

Figure 4-9 Typical Relations of Estimated Efforts VS Actual Efforts Using Reduced Multiple Regression With Optimal Subset of 9 Inputs Guided by GA

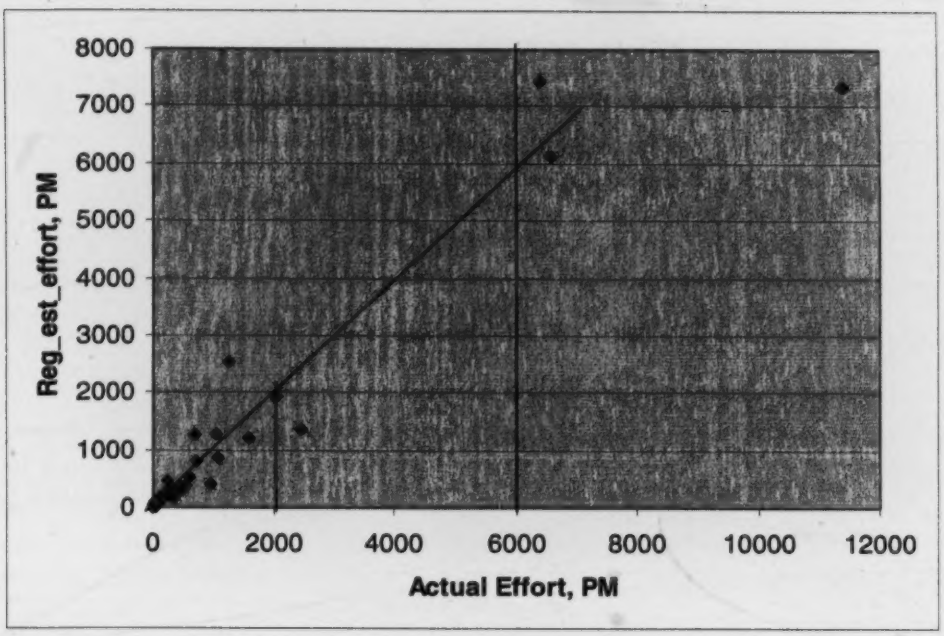

Figure 4-10 Typical Relations of Estimated Efforts VS Actual Efforts Using Full Multiple Regression With 16 Explanatory Variables

4.3.2 Using Neural Networks to Evaluate Fitness

Note that RMSE* in Table 4-2 is the results obtained from separately using neural networks with take-one-out cross validation in 3000 epochs, not from GA experiments, which are lower in GA since GA uses the network training RMSE in stead of using take-one-out cross validation.

From Table 4-2, it is clear that the subset of 8 variables selected as optimal members are different, these subsets are local optimal subsets but it is hard to tell if they are also globally optimal subsets. This phenomenon reveals that even the GA cannot locate the globally optimal subset of variables, this is because the evaluation of fitness by using neural networks is not repeatable. This slightly unstable performance of neural networks prevents the GA's ability from locating the globally optimal subset.

Table 4-2. Optimal Subset of Variables by GA Using NN to Evaluate the Fitness

Subset Label	GA Parameters	Optimal Subset	RMSE*	Pred (25), %	MMRE, %
Linear	NONE	1,3,4,8,10,11,14,15	0.0316	76.19	18.75
GA-952	CR=0.95, MR=0.2	1,3,4,5,8,11,12,15	0.0359	65.08	22.90
GA-951	CR=0.95, MR=0.1	2,3,5,6,9,10,11,12	0.0464	60.32	28.50
GA-9501	CR=0.95, MR=0.01	6,7,9,11,12,13,14,15	0.0543	63.49	31.01
GA-852	CR=0.85, MR=0.2	1,2,4,5,6,7,9,12	0.0506	53.97	36.05
GA-851	CR=0.85, MR=0.1	2,4,7,8,10,11,12,14	0.0408	65.08	23.10
GA-8501	CR=0.85, MR=0.01	2,3,4,5,7,10,14,15	0.0461	58.73	28.80
GA-752	CR=0.75, MR=0.2	3,4,6,7,8,10,11,15	0.0394	65.08	24.85
GA-751	CR=0.75, MR=0.1	1,3,4,6,7,10,11,14	0.0372	63.49	23.27
GA-7501	CR=0.75, MR=0.01	2,6,7,8,10,11,12,14	0.0652	50.79	42.49

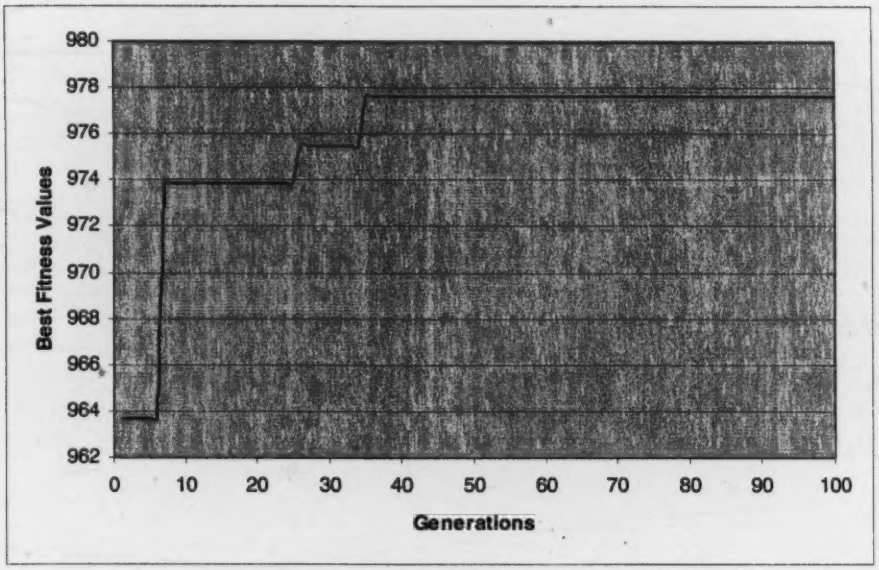

Figure 4-11Typical Relations of Best Fitness VS Generations

by GA Using Neural Networks as Fitness Function

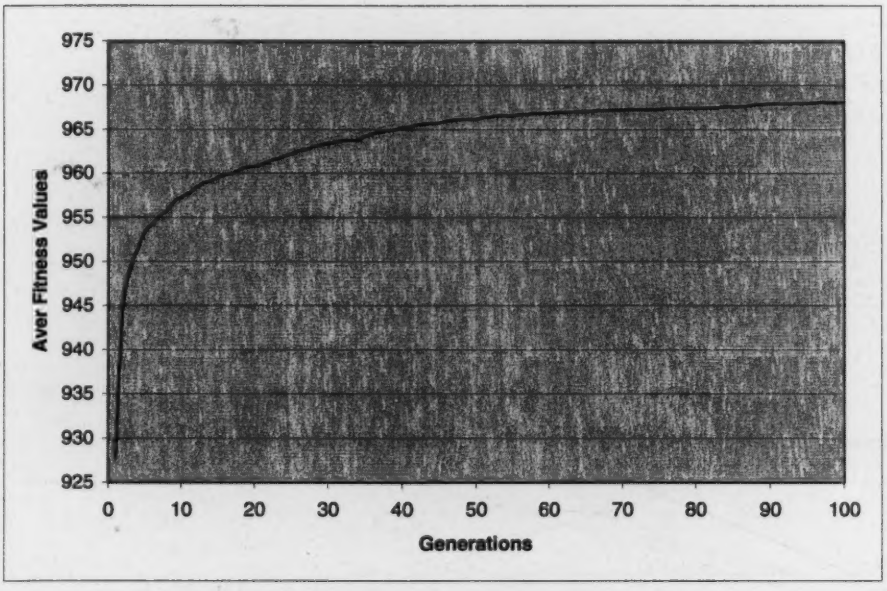

Figure 4-12 Typical Relations of Average Fitness VS Generations

by GA Using Neural Networks as Fitness Function

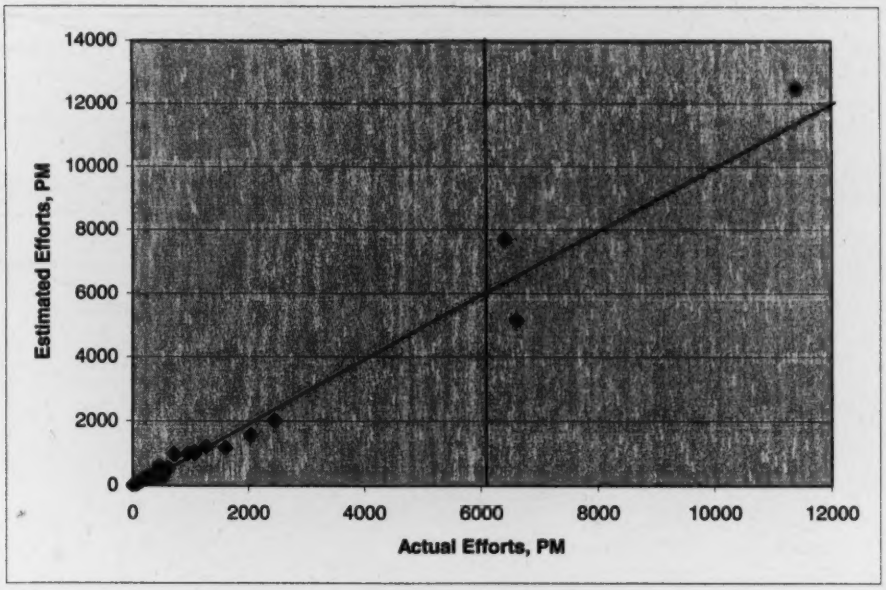

Figure 4-13 Typical Relations of Estimated Efforts VS Actual Efforts Using Reduced NN Simulation With Optimal Subset of 9 Inputs Found by GA-NN

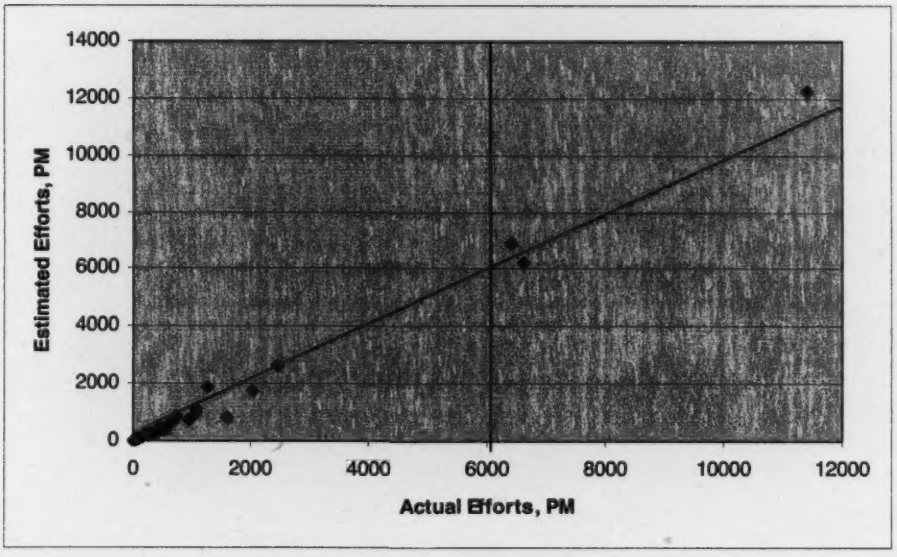

Figure 4-14 Typical Relations of Estimated Efforts VS Actual Efforts Using Reduced NN Simulation With Optimal Subset of 9 Inputs Obtained by GA-Multiple Linear Regression

It can also be seen that the typical subset's performance in the reduced neural network simulation in Figure 4-13 is not good as that in Figure 4-14, this indicates that the optimal subset obtained by linear techniques in Chapter 2 is closer to the globally optimal subset. It is obvious that it is the NN's unstable performance that causes the GA-NN's inability to locate the globally optimal subset, such phenomenon was not found in GA-Multiple regression experiments. To solve this problem, replace the neural networks with multiple linear regression, whose fitness evaluation is completely repeatable when applied to the same data. The results are listed in Table 4-1. When comparing the performance of the GA using multiple regression with NN to evaluate the fitness in Figure 4-7, Figure 4-8, Figure 4-11 and Figure 4-12, respectively, it can be found that their GA's performance is very similar: the best and the average fitness values increase with generations, this observation is reasonable and is expected for the GA.

4.3.3 Comparisons of Linear Methods and Genetic Algorithms on Feature Optimization

Variable selection procedures such as forward selection, backward elimination, stepwise selection, Maximum R^2 improvement, and Minimum R^2 improvement introduced in Chapter 2 are straightforward and efficient in identifying the most important factors if the number of original variables in full models is not too big, further more, it needs the users to apply intelligent constraints to find the globally optimal subset, and it is not guaranteed that all linear search techniques are able to find the globally optimal subset because of their limitations. For example, forward selection and stepwise selection can only find the local optimal subset. Regarding the GA for feature optimization, since it always keeps the best chromosomes during search optimal subset of features, and produces optimal subsets

randomly and not constrained to linear directions, it is more efficient when the input space is complex and the number of explanatory factors is large, sooner or later, the globally optimal chromosome can be found if the fitness function is repeatable.

Table 4-3 and Table 4-4 listed the typical effort estimation and their performances by various models, the experimental results show that the NN simulation outperforms the regression models. Flexibility, objectivity, correctness and computational economy are desirable features that make neural networks attractive as a learning-oriented estimator for software development effort. However, using NN to evaluate the fitness in GA is not suitable because of its slightly unstable performance, while multiple regression is repeatable to evaluate the GA fitness and it is suitable to be hired by GA to evaluate the fitness.

Table 4-3. Typical Effort Estimation Using Different Models

Software Project #	Actual Effort, PM	COCOMO'81 Intermediate Estimation	Full Neural Networks Simulation	Reduced Neural Networks Simulation	Full Multiple Linear Regression	Reduced Multiple Linear Regression
18	11400	11056	8150.67	11595.83	7348.65	5815.92
19	6600	7764	6377.42	9906.47	6100.41	6669.29
20	6400	6536	6963.04	6495.71	7435.68	6527.79
21	2455	1836	2803.28	1633.86	1362.62	1262.69
1	2040	2218	2028.97	1902.69	1925.07	1490.15
2	1600	1770	1895.53	1271.36	1213.75	930.94
31	1063	962	1365.11	1347.63	1252.80	1193.68
56	958	537	1006.71	838.40	396.65	505.99
22	724	733	867.56	866.60	804.52	953.67
33	605	529	496.54	409.25	539.39	403.94
25	523	430	470.55	523.10	327.96	326.97
9	423	397	377.52	420.76	299.75	312.94
26	387	339	299.84	337.03	232.78	236.17
34	230	201	239.65	231.87	251.83	280.51
50	176	193	162.29	157.97	162.55	178.65
28	98	133	93.75	89.50	131.25	97.50
44	87	130	77.87	77.08	106.09	94.94
14	73	60	71.60	74.79	64.38	58.85
15	61	52	57.66	59.71	93.74	111.66
36	55	33	60.64	55.11	50.96	51.53
6	43	30	43.06	55.07	31.98	30.95
47	36	33	35.70	43.78	32.53	39.08
54	20	14	19.48	19.62	13.65	15.14
55	18	7.5	17.97	16.05	7.13	7.62
7	8	9.8	7.89	6.61	8.48	8.19

Table 4-4. Comparison of Performance of Different Models

Model	Full Regression	Reduced Regression	Full NN	Reduced NN	COCOMO'81 Intermediate
MMRE, %	30.36	33.88	8.94	17.94	18.74
Pred (25), %	58.73	55.56	90.48	74.60	74.68

4.4 Summary

The reduced multiple regression models for effort estimation proposed in this thesis are straightforward and much simpler to use, with only slightly lower prediction accuracy

than that obtained using COCOMO'81 intermediate model. The variable selection procedures such as forward selection, backward elimination, stepwise selection, Maximum R^2 improvement, and Minimum R^2 improvement are efficient in identifying the most significant factors contributing to software development effort. Therefore, it is especially important for managers to be aware of these factors in making decisions in the early stage of software development.

The experimental results show that the NN simulation outperforms the regression models for effort estimation. Flexibility, objectivity, correctness and computational economy are desirable features that make neural networks attractive as a learning-oriented estimator for software development effort. However, using NN to evaluate the fitness in GA is not suitable because of NN's slightly unstable performance, multiple regression is more suitable as a tool to evaluate the GA fitness.

4.5.Bibliography

[1] Michalewicz, Z., Genetic Algorithms + Data Structures = Evolution Programs, Third, Revised and Extended Edition, Springer,1999.

[2] M. Mitchell, *An introduction to Genetic Algorithms*, The MIT press, 1996

[3] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

[4] M. Mitchell, J.H. Holland, and S. Forrest, "When will a genetic algorithm outperform hill climbing?," In *Advances in Neural Information Processing Systems 6*, Morgan Kaufmann, 1994.

Chapter 5 Conclusions and Future Work

5.1 Conclusions

There are three main contributions of this thesis:

It has thoroughly investigated the performance of multiple regression models and neural networks in effort estimation. It has successfully built efficient reduced models providing high predicting accuracy. It has identified the most significant factors among the original set of independent variables.

5.1.1 Multiple Regression Models and Reduced Models

The full regression model is successfully built using the available COCOMO dataset with 16 independent variables related to the effort estimation. The full regression model is more straightforward and simple to use and its performance is compatible to that of the COCOMO'81 intermediate model, with only slightly lower prediction accuracy .

To build more efficient models for effort estimation, the traditional linear variable selection procedures such as forward selection, backward elimination, stepwise selection, Maximum R^2 improvement, and Minimum R^2 improvement are employed and they are found to be efficient in identifying the most significant factors contributing to software development effort. The performance of the optimal reduced regression model is very close to that of the full regression model.

Experimental results also showed that there are several potential outliers for both the full regression model and the reduced models, but their effects are not significant and thus can be ignored.

5.1.2 Neural Networks Simulations

Performance of the neural networks with unipolar sigmoid in hidden layer and linear function in output layer as transfer functions is affected by the investigated factors: learning rate constants, the number of hidden nodes, momentum, and the lambda values. Three factors: learning rate constants, momentum and lambda values, can be instrumental to speed up the learning processes and the RMSE convergence, but it is advisable to control only one of them and let the other two factors be in lower or normal levels.

When using the bipolar sigmoid function as transfer function in both hidden layer and output layer, the optimal parameter combinations found are: epoch = 3000, momentum = 0.1, the number of hidden nodes = 20, learning rate = 0.05, λ = 1.0. For the neural networks with the unipolar sigmoid function as transfer function in both hidden layer and output layer, the optimal parameter combinations are: epoch = 3000, momentum = 0.1, number of hidden nodes = 20, learning rate = 0.95, λ = 1.0. The learning processes of both networks are stable under the optimal parameter combinations. When the number of hidden nodes is reduced to 5, the optimal parameter combination is: epoch = 3000, momentum = 0.1, learning rates = 0.020, λ = 1.0. Based on the optimal parameters combination of the networks with unipolar sigmoid function in the hidden layer and the linear function in the output layer, the networks with categorical input variables outperform the ones with continuous numerical input variables.

The experimental results showed that the neural networks used as effort estimators outperform the regression models for effort estimation. Flexibility, objectivity, correctness and computational economy are desirable features that make neural networks

attractive as a learning-oriented estimator for software development effort. However, optimal parameter combinations of neural networks have to be found, and the simulations are less repeatable.

5.1.3 Feature Selection and Optimization Using Linear and Non-linear Search Methods

Building more efficient models and identifying the most significant factors among variables are the focuses in this thesis. Traditional linear techniques and genetic algorithms are employed for the feature set selection of the potential significant variables. Linear techniques are straightforward but less efficient when the number of dataset attributes is large. Genetic algorithms are more efficient for feature set selection, and thus more powerful to build the optimal reduced models.

The nine most important factors identified in this globally optimal model in decreasing order are: ln_size, VEXP, ACAP, RELY, TIME, PCAP, SCED, TOOL and CPLX. Those factors belong to various categories of attributes in the COCOMO dataset.

The interesting findings in this thesis are that human capabilities in project management are the most important, followed by project attributes and product attributes, while the computer attributes contribute least in determining the software development effort. Therefore, it is especially important for managers to be aware of these factors in making decisions in the early stage of software development.

5.2 Future Work

Building More Efficient Models. The models in this step should be more capable of preprocessing the collected data, ensures the data reliability of data and results and avoids the unstable performance in effort estimation, shorten the learning time in neural networks training.

Building More Interpretable Models. The models should be able to produce a set of rules that are clear to understand and use. The relationships among the attributes in the models should be clearly explained and the effort estimation is repeatable based on the high accuracy. The models should be integrated for data preprocessing, data mining and highly intelligent and automated for users.

Appendices

Appendix 1 COCOMO Dataset with 63 Projects and 17 Dimensions

	Var 1	Var 2	Var 3	Var 4	Var 5	Var 6	Var 7	Var 8	Var 9	Var 10	Var 11	Var 12	Var 13	Var 14	Var 15	Var 16	Var 17	
Proj #	RE -LY	DA -TA	CP -LX	TI -ME	ST -OR	VI -RT	TU -RN	AC -AP	AE -XP	PC -AP	VE -XP	LE -XP	MO -DP	TO -OL	SC -ED	Soft -ware Size	ACT Effort	EAF Target
1	0.88	1.16	0.7	1	1.06	1.15	1.07	1.19	1.13	1.17	1.1	1	1.24	1.1	1.04	113	2040	2.505
2	0.88	1.16	0.85	1	1.06	1	1.07	1	0.91	1	0.9	0.95	1.1	1	1	293	1600	0.626
3	1	1.16	0.85	1	1	0.87	0.94	0.86	0.82	0.86	0.9	0.95	0.91	0.91	1	132	243	0.342
4	0.75	1.16	0.7	1	1	0.87	1	1.19	0.91	1.42	1	0.95	1.24	1	1.04	60	240	1.019
5	0.88	0.94	1	1	1	0.87	1	1	1	0.86	0.9	0.95	1.24	1	1	16	33	0.561
6	0.75	1	0.85	1	1.21	1	1	1.46	1	1.42	0.9	0.95	1.24	1.1	1	4	43	3.134
7	0.75	1	1	1	1	0.87	0.87	1	1	1	0.9	0.95	0.91	0.91	1	6.9	8	0.329
8	1.15	0.94	1.3	1.66	1.56	1.3	1	0.71	0.91	1	1.21	1.14	1.1	1.1	1.08	22	1075	9.405
9	1.15	0.94	1.3	1.3	1.21	1.15	1	0.86	1	0.86	1.1	1.07	0.91	1	1	30	423	2.551
10	1.4	0.94	1.3	1.11	1.56	1	1.07	0.86	0.82	0.86	0.9	1	1	1	1	29	321	2.016
11	1.4	0.94	1.3	1.11	1.56	1	1.07	0.86	0.82	0.86	0.9	1	1	1	1	32	218	1.217
12	1.15	0.94	1.3	1.11	1.06	1	1	0.86	0.82	0.86	1	0.95	0.91	1	1.08	37	201	0.942
13	1.15	0.94	1.3	1.11	1.06	1.15	1	0.71	1	0.7	1.1	1	0.82	1	1	25	79	0.593
14	1.15	0.94	1.65	1.3	1.56	1.15	1	0.86	1	0.7	1.1	1.07	1.1	1.24	1.23	3	73	7.109
15	1.4	0.94	1.3	1.3	1.06	1.15	0.87	0.86	1.13	0.86	1.21	1.14	0.91	1	1.23	3.9	61	4.253
16	1.4	1	1.3	1.3	1.56	1	0.87	0.86	1	0.86	1	1	1	1	1	6.1	40	1.631
17	1.4	1	1.3	1.3	1.56	1	0.87	0.86	0.82	0.86	1	1	1	1	1	3.6	9	0.691
18	1.15	1.16	1.15	1.3	1.21	1	1.07	0.86	1	1	1	1	1.24	1.1	1.08	320	11400	4.014
19	1.15	1.08	1	1.11	1.21	0.87	0.94	0.71	0.91	1	1	1	0.91	0.91	1	1150	6600	0.501
20	1.4	1.08	1.3	1.11	1.21	1.15	1.07	0.71	0.82	1.08	1.1	1.07	1.24	1	1.08	299	6400	3.600
21	1	1.16	1.15	1.06	1.14	0.87	0.87	0.86	1	1	1	1	0.91	0.91	1	252	2455	1.151
22	1.15	1	1	1.27	1.06	1	1	0.86	0.82	0.86	0.9	1	0.91	1	1.23	118	724	0.844
23	1.15	1	1	1.08	1.06	1	1	0.86	0.82	0.86	0.9	1	1	1	1.23	77	539	1.049
24	0.88	1	0.85	1.06	1.06	1	0.87	1	1.29	1	1.1	0.95	0.82	0.83	1	90	453	0.978
25	1.15	1.16	1.3	1.15	1.06	1	0.87	0.86	1	0.86	1.1	1	0.82	0.91	1.08	38	523	2.375
26	0.94	1	0.85	1.07	1.06	1.15	1.07	0.86	1	0.86	1.1	1	0.91	1.1	1.08	48	387	1.328
27	1.15	0.94	1.15	1.35	1.21	1	0.87	1	1	1	1	1	0.82	1.1	1.08	9.4	88	2.136
28	1.15	1.08	1.3	1.11	1.21	1.15	1.07	0.86	1	0.86	1.1	1.07	1.1	1.1	1	13	98	2.072
29	0.88	1	1	1	1	1	1	1.1	1.29	0.86	1	1	0.91	0.91	1.23	2.14	7.3	1.046
30	0.88	1	1	1	1	1	1	1	1.29	0.86	1	1	0.91	0.91	1.23	1.98	5.9	0.928
31	1.4	1.08	1	1.48	1.56	1.15	1.07	0.86	0.82	0.86	1.1	1.07	1	1	1	62	1063	2.682
32	0.88	1.08	0.85	1	1	1	1	0.71	0.82	1	1	1	1.1	1.1	1	390	702	0.293
33	1.4	1.08	1.3	1.48	1.56	1.15	0.94	0.86	0.82	0.86	0.9	1	0.91	0.91	1	42	605	2.436
34	1.15	1.08	1	1.06	1	1	0.87	1	1	1	1	1	0.91	1.1	1.23	23	230	1.908
35	0.75	0.94	1.3	1.06	1.21	1.15	1	1	0.91	1	1.1	1	1.24	1.24	1	13	82	1.349
36	0.88	1.08	0.85	1	1	0.87	0.87	1.19	1	1.17	0.9	0.95	1	0.91	1.04	15	55	0.883
37	0.88	0.94	0.7	1	1.06	1	1	0.86	0.82	0.86	1	1	1	1	1	60	47	0.200
38	1	1	1.15	1	1	0.87	0.87	0.71	0.91	1	0.9	0.95	0.82	0.91	1	15	12	0.218
39	1	1	1.15	1	1	0.87	1	0.71	0.82	0.7	1	0.95	0.91	1.1	1	6.2	8	0.368
40	1	0.94	1.3	1	1	1	0.87	0.86	0.82	1.17	1	1	1.1	1	1	3	8	0.789

41	0.88	0.94	1	1	1	0.87	0.87	1	0.82	0.7	0.9	0.95	0.91	0.91	1	5.3	6	0.326
42	0.88	1.04	1.07	1	1.06	0.87	1.07	0.86	1	0.93	0.9	0.95	0.95	0.95	1.04	45.5	45	0.255
43	1	1.04	1.07	1	1.21	0.87	1.07	0.86	1	1	0.9	0.95	1	1	1.04	28.6	83	0.770
44	0.88	1.04	1.07	1.06	1.21	0.87	1.07	1	1	1	0.9	0.95	1.1	1	1.04	30.6	87	0.749
45	0.88	1.04	1.07	1	1.06	0.87	1.07	1	1	1	0.9	0.95	1	0.95	1.04	35	106	0.792
46	0.88	1.04	1.07	1	1.06	0.87	1.07	1	1	0.86	0.9	0.95	1	1	1.04	73	126	0.435
47	0.75	0.94	1.3	1	1	0.87	0.87	0.71	0.82	0.7	1.1	1.07	1.1	1	1.04	23	36	0.418
48	0.88	0.94	0.85	1	1	0.87	1	1.19	0.91	1.17	0.9	0.95	1.1	1	1.04	464	1272	0.437
49	1	1	0.85	1	1	1	0.87	0.71	1	0.7	1.1	1	0.82	0.91	1	91	156	0.333
50	1.15	1	1	1.3	1.21	1	0.87	0.86	1	0.86	1.1	1	1	1	1	24	176	1.387
51	0.88	1	1	1	1	1	1.15	1.19	1	1.42	1	0.95	1.24	1.1	1.04	10	122	3.398
52	0.88	0.94	0.85	1	1.06	1.15	1	1	1	1	1.1	1.07	1.24	1.1	1	8.2	41	1.407
53	0.88	0.94	1.15	1.11	1.21	1.3	1	0.71	1	0.7	1.1	1.07	1	1.1	1.08	5.3	14	0.721
54	1	0.94	1	1	1.06	1.15	0.87	1	0.82	1	1	0.95	0.91	1.1	1	4.4	20	1.319
55	0.88	0.94	0.7	1	1	0.87	0.87	0.86	0.82	1.17	0.9	0.95	1.1	1	1	6.3	18	0.814
56	1.15	0.94	1.3	1.3	1.21	1	1	0.86	0.91	1	1.1	1.07	1.1	1.1	1.08	27	958	6.555
57	1	0.94	1.15	1.11	1.21	1.3	1	1	1	1	1.1	1.07	1.1	1.1	1.23	17	237	2.825
58	1.4	0.94	1.3	1.66	1.21	1	1	0.71	0.82	0.7	0.9	0.95	0.91	1	1	25	130	0.976
59	1	0.94	1.15	1.06	1.06	1	0.87	1	1	1	1	1	0.91	1	1	23	70	0.813
60	1.15	0.94	1.3	1.11	1.06	1	1	0.86	1.13	0.86	1.1	1.07	1.1	1.1	1.08	6.7	57	2.417
61	1	0.94	1.15	1	1	0.87	0.87	0.86	1	0.86	0.9	1	0.82	1	1	28	50	0.472
62	0.88	0.94	1.3	1.11	1.21	1.15	1	0.78	0.82	0.7	1.21	1.14	0.91	1.24	1	9.1	38	1.068
63	1	0.94	1.15	1	1	1	0.87	0.71	0.82	0.86	1	1	0.82	1	1	10	15	0.338

Appendix 2. Attribute Number, Name and Their Meanings

Attribute #	**Cost Drivers**
	Product Attributes
1	*RELY*: Required Software Reliability
2	*DATA*: Data Base Size
3	*CPLX*: Product Complexity
	Computer Attributes
4	*TIME*: Execution Time Constraint
5	*STOR*: Main Storage Constraint
6	*VIRT*: Virtual Machine Volatility
7	*TURN*: Computer Turnaround Time
	Personnel Attributes
8	*ACAP*: Analyst Capability
9	*AEXP*: Applications Experience
10	*PCAP*: Programmer Capability
11	*VEXP*: Virtual Machine Experience
12	*LEXP*: Program Language Experience
	Project Attributes
13	*MODP*: Modern Programming Practices
14	*TOOL*: Use of Software Tools
15	*SCED*: Development Schedule

Appendix 3. Categorical Data of 15 Input Variables in Appendix 3

	Var 1	Var 2	Var 3	Var 4	Var 5	Var 6	Var 7	Var 8	Var 9	Var 10	Var 11	Var 12	Var 13	Var 14	Var 15	Var 16	Var 17	
Proj #	RE -LY	DA -TA	CP -LX	TI -ME	ST -OR	VI -RT	TU -RN	AC -AP	AE -XP	PC -AP	VE -XP	LE -XP	MO -DP	TO -OL	SC -ED	Soft -ware Size	ACT Effort	EAF Target
1	2	4	1	1	2	3	3	2	2	2	2	3	1	2	4	113	2040	2.505
2	2	4	2	1	2	2	3	3	4	3	4	4	2	3	3	293	1600	0.626
3	3	4	2	1	1	1	1	4	4	4	4	4	4	4	3	132	243	0.342
4	1	4	1	1	1	1	2	2	4	1	3	4	1	3	4	60	240	1.019
5	2	1	3	1	1	1	2	3	3	4	4	4	1	3	3	16	33	0.561
6	1	2	2	1	3	2	2	1	3	1	4	4	1	2	3	4	43	3.134
7	1	2	3	1	1	1	1	3	3	3	4	4	4	4	3	6.9	8	0.329
8	4	1	5	4	4	4	2	5	4	3	1	1	2	2	2	22	1075	9.405
9	4	1	5	3	3	3	2	4	3	4	2	2	4	3	3	30	423	2.551
10	5	1	5	2	4	2	3	4	5	4	4	3	3	3	3	29	321	2.016
11	5	1	5	2	4	2	3	4	5	4	4	3	3	3	3	32	218	1.217
12	4	1	5	2	2	2	2	4	5	4	3	4	4	3	2	37	201	0.942
13	4	1	5	2	2	3	2	5	3	5	2	3	5	3	3	25	79	0.593
14	4	1	6	3	4	3	2	4	3	5	2	2	2	1	1	3	73	7.109
15	5	1	5	3	2	3	1	4	2	4	1	1	4	3	1	3.9	61	4.253
16	5	2	5	3	4	2	1	4	3	4	3	3	3	3	3	6.1	40	1.631
17	5	2	5	3	4	2	1	4	5	4	3	3	3	3	3	3.6	9	0.691
18	4	4	4	3	3	2	3	4	3	3	3	3	1	2	2	320	11400	4.014
19	4	3	3	2	3	1	1	5	4	3	3	3	4	4	3	1150	6600	0.501
20	5	3	5	2	3	3	3	5	5	2	2	2	1	3	2	299	6400	3.600
21	3	4	4	2	3	1	1	4	3	3	3	3	4	4	3	252	2455	1.151
22	4	2	3	3	2	2	2	4	5	4	4	3	4	3	1	118	724	0.844
23	4	2	3	2	2	2	2	4	5	4	4	3	3	3	1	77	539	1.049
24	2	2	2	2	2	2	1	3	1	3	2	4	5	5	3	90	453	0.978
25	4	4	5	2	2	2	1	4	3	4	2	3	5	4	2	38	523	2.375
26	2	2	2	2	2	3	3	4	3	4	2	3	4	2	2	48	387	1.328
27	4	1	4	3	3	2	1	3	3	3	3	3	5	2	2	9.4	88	2.136
28	4	3	5	2	3	3	3	4	3	4	2	2	2	2	3	13	98	2.072
29	2	2	3	1	1	2	2	2	1	4	3	3	4	4	1	2.14	7.3	1.046
30	2	2	3	1	1	2	2	3	1	4	3	3	4	4	1	1.98	5.9	0.928
31	5	3	3	4	4	3	3	4	5	4	2	2	3	3	3	62	1063	2.682
32	2	3	2	1	1	2	2	5	5	3	3	3	2	2	3	390	702	0.293
33	5	3	5	4	4	3	1	4	5	4	4	3	4	4	3	42	605	2.436
34	4	3	3	2	1	2	1	3	3	3	3	3	4	2	1	23	230	1.908
35	1	1	5	2	3	3	2	3	4	3	2	3	1	1	3	13	82	1.349
36	2	3	2	1	1	1	1	2	3	2	4	4	3	4	4	15	55	0.883
37	2	1	1	1	2	2	2	4	5	4	3	3	3	3	3	60	47	0.200
38	3	2	4	1	1	1	1	5	4	3	4	4	5	4	3	15	12	0.218
39	3	2	4	1	1	1	2	5	5	5	3	4	4	2	3	6.2	8	0.368

40	3	1	5	1	1	2	1	4	5	2	3	3	2	3	3	3	8	0.789
41	2	1	3	1	1	1	1	3	5	5	4	4	4	4	3	5.3	6	0.326
42	2	3	4	1	2	1	3	4	3	4	4	4	4	4	4	45.5	45	0.255
43	3	3	4	1	3	1	3	4	3	3	4	4	3	3	4	28.6	83	0.770
44	2	3	4	2	3	1	3	3	3	3	4	4	2	3	4	30.6	87	0.749
45	2	3	4	1	2	1	3	3	3	3	4	4	3	4	4	35	106	0.792
46	2	3	4	1	2	1	3	3	3	4	4	4	3	3	4	73	126	0.435
47	1	1	5	1	1	1	1	5	5	5	2	2	2	3	4	23	36	0.418
48	2	1	2	1	1	1	2	2	4	2	4	4	2	3	4	464	1272	0.437
49	3	2	2	1	1	2	1	5	3	5	2	3	5	4	3	91	156	0.333
50	4	2	3	3	3	2	1	4	3	4	2	3	3	3	3	24	176	1.387
51	2	2	3	1	1	2	4	2	3	1	3	4	1	2	4	10	122	3.398
52	2	1	2	1	2	3	2	3	3	3	2	2	1	2	3	8.2	41	1.407
53	2	1	4	2	3	4	2	5	3	5	2	2	3	2	2	5.3	14	0.721
54	3	1	3	1	2	3	1	3	5	3	3	4	4	2	3	4.4	20	1.319
55	2	1	1	1	1	1	1	4	5	2	4	4	2	3	3	6.3	18	0.814
56	4	1	5	3	3	2	2	4	4	3	2	2	2	2	2	27	958	6.555
57	3	1	4	2	3	4	2	3	3	3	2	2	2	2	1	17	237	2.825
58	5	1	5	4	3	2	2	5	5	5	4	4	4	3	3	25	130	0.976
59	3	1	4	2	2	2	1	3	3	3	3	3	4	3	3	23	70	0.813
60	4	1	5	2	2	2	2	4	2	4	2	2	2	2	2	6.7	57	2.417
61	3	1	4	1	1	1	1	4	3	4	4	3	5	3	3	28	50	0.472
62	2	1	5	2	3	3	2	5	5	5	1	1	4	1	3	9.1	38	1.068
63	3	1	4	1	1	2	1	5	5	4	3	3	5	3	3	10	15	0.338